'68' MICRO JOURNAL

$2.95USA

Australia	A $ 4.75	New Zealand	NZ $ 4.75
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30:-SEK

VOLUME IV ISSUE VI • Devoted to the 68XX User • June 1982

"Small Computers Doing Big Things"

YOUR CHOICE-smart either way

- Over 140 software driven functions
- 82 x 24 or 82 x 20 screen format — software selectable
- High resolution 7 x 12 matrix characters — P-31 green phosphor
- Upper/lower case character set — plus graphics character set
- 56-key alphanumeric keyboard — plus 12-key cursor, numeric pad
- Internal editing functions — insert, delete, scroll, roll, slide, etc.
- Parallel printer I/O port
- 50 to 38,400 baud operation — programmable
- Cursor type, cursor position, print control characters, protected fields, shift inversion, dual intensity and many other features

8212 — twelve-inch diagnonal screen or **8209** — nine-inch diagnonal screen

SWTP

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Pascal for 6809

Pascal for the 6809 is a true native code compiler. Unlike the usual P-code Pascals which run in an interpretive manner, ours produces efficient assembly language mnemonics which can be assembled and run directly. This compiler is available for both 6809 FLEX™ and UniFLEX™. Many features not found in other Pascal systems were implemented while avoiding those features completely non-standard. Features of the Pascal system include:

- Supports most of Jensen and Wirth specification
- Produces fast and efficient 6809, native code
- FLEX run-time package may be trimmed
- Double precision real numbers (16.8 digits)
- Implements scalar, subrange and structured data types
- Standard I/0 using file buffer pointers
- Dynamic storage allocation
- Ability to call other Pascal programs
- FLEX version may call assembly language programs
- Buffered or single character terminal input
- Standard math functions: SIN, COS, ARCTAN, EXP, LN, SQR, SQRT
- Random number generator function
- Many usable, sample programs included
- UniFLEX version supports:
  - Random file positioning
  - Ability to call various UniFLEX system routines
  - Ability to execute UniFLEX utility commands

Pascal on diskette for 5" and 8" 6809 FLEX is available for $200.00 The 5" version requires two disk drives. The UniFLEX version is $300.00 and includes one year of maintenance. All orders should include 3 percent for postage and handling (10 percent on foreign orders).

™FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.

technical systems consultants, inc.

111 Providence Rd., Chapel Hill, N.C. 27514

(919) 493-1451

'68'

MICRO JOURNAL

Portions of text prepared using the following.

SWTPC 6800-6809-DMAF2-CDS1-CT82-Sprint 3
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216

EDITOR - WORD PROCESSOR
Technical Systems Consultants, Inc.
Box 2573, W. Lafayette, IN 47906
FLEX is TM of TSC

GIMIX Super Mainframe-Assorted memory boards
GIMIX Inc.
1337 West 37th Place
Chicago, Il 60609

TABLE OF CONTENTS

Vol. IV, Issue VI June '82

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68 Micro Journal is published 12 times a year by Computer Publishing, Inc. Second class postage paid at Hixson, Tennessee and additional entries. Postmaster: send Form 3579 to 68 Micro Journal, PO Box 849, Hixson, Tennessee.

Subscription Rates

1-Year $24.50 2-Years $42.50 3-Years $64.50 USA

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors full name, address, date and telephone number. It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands ONLY should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single speced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

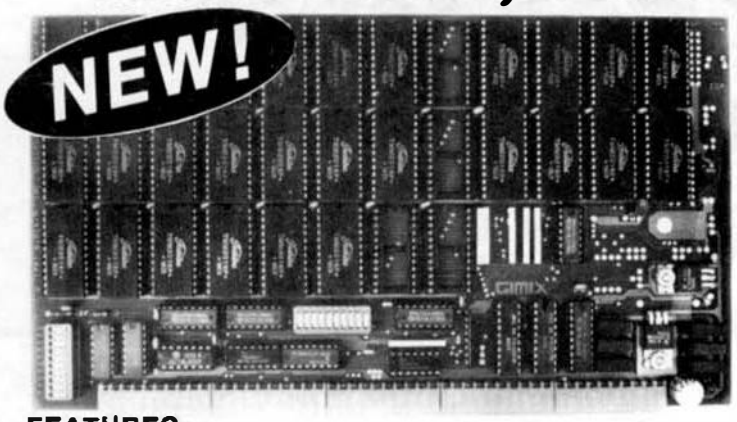
All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well es 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste',

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

64K BYTE STATIC RAM BOARD $638.67

for 6800 and 6809 systems using the SS-50/SS-50C bus



also available:

56K $578.57
48K $518.47
32K $398.37
24K $348.27

All versions have gold bus connectors and are fully socketed, assembled, burned in, and tested. Versions with less than 64K can be expanded at any time by adding additional RAM chips.

FEATURES:

★ ADDRESSABLE in two 32K sections with separate regular and extended address decoding for each section. Each section can be addressed to any 32K boundary in the address range (1M Byte with extended addressing). Each 32K section is divided into four 8K blocks that can be individually enabled or disabled. Disabled sections do not occupy address space.

★ FULLY STATIC MEMORY does not require complicated refresh timing or clocks for data retention. Compatible with any of the 6800/6809 DMA techniques.

★ GUARANTEED 2Mhz. OPERATION uses high speed (200 ns.) memory with no wait states or clock stretching required.

★ LOW POWER NMOS RAM requires less than 3/4 AMP (750 ma) typical at 8V, for a fully populated 64K board.

Also Available:

NON-VOLATILE 64K BYTE CMOS STATIC RAM BOARDS with BATTERY BACK-UP

With all the versatility of the above boards -- PLUS!

★ NON-VOLATILE MEMORY with built in battery back-up. Retains data even with system power removed. With the battery fully charged, data remains intact for a minimum of 21 days.

★ ULTRA-LOW POWER CMOS RAM requires less than 1/4 AMP (250 ma.) typical at 8V for a fully populated 64K board.

★ LOW BUS VOLTAGE DETECTION inhibits memory access during power up and power down to prevent false writes to the memory.

★ WRITE PROTECT SWITCH permits the entire board to be write protected for PROM/ROM emulation and software debugging.

64K.....$798.64 **56K.....$728.56** **32K.....$518.36**

GIMIX UNIVERSAL SYNCHRONOUS & ASYNCHRONOUS SERIAL I/O BOARDS. This 30 pin board is available in three versions; with a **68B50 ACIA**, a **68B52 SSDA** (Synchronous Serial Data Adapter) or a **68B54 ADLC** (Advanced Data-Link Controller). Control logic is provided for loop mode operation of the 68B54 ADLC. All three feature jumper selectable **RS-232C** or **RS-423** (single-ended), or **RS-422** (Differential) line drivers and receivers for the Receive data, transmit data, external clock, and handshake signals. External connections can be made through the 26 pin header at the top of the board or, when used with an optinal GIMIX cable set, a 25 pin "D" type data connector. The jumper programmable I/O connector pinouts can be arranged to suit a variety of interface configurations.

68B50 ACIA ($244.50)
68B52 SSDA ($254.52)
68B54 ADLC ($268.54)

TO ORDER BY MAIL

SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 6 weeks to collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Commercial Illinois National Bank of Chicago, account #73-32033. Visa or Master Charge also accepted.

GIMIX INC. reserves the right to change pricing and product specifications at any time without further notice.

GIMIX™ and GHOST™ are registered trademarks of GIMIX Inc. 1981 GIMIX Inc.

FLEX AND Uniflex are trademarks of Technical Systems Consultants Inc. OS-9 is a trademark of Microware Inc. See their ads for other GIMIX compatible software.

GIMIX INC.

The Company that delivers
Quality Electronic products since 1975.

1337 WEST 37th PLACE, CHICAGO, IL 60609
(312) 927-5510 • TWX 910-221-4055

Build performance into your system using the OS-9™ toolbox!

Microware's OS-9 Toolbox holds the finest 6809 programming languages and software tools in the industry. Microware languages are complete language implementations that meet or exceed applicable industry standards, and have unmatched speed, efficiency, and ease of use that Microware software is famous for. *All Microware languages run on OS-9 Levels One or Two.*

OS-9 PASCAL™ LANGUAGE COMPILER

The OS-9 Pascal language compiler is the most complete and versatile Pascal available for the 6809. OS-9 Pascal has the unusual capability of generating P-code for interpretive execution while debugging OR highly optimized 6809 assembly language source code output for maximum speed. Another feature of OS-9 Pascal is its "virtual memory" P-code interpreter that lets you run incredibly large Pascal programs. OS-9 Pascal meets the ISO 7185.1 Standard and the complete Wirth/Jensen specification.

CIS COBOL™ COMPILER

6809 CIS Cobol is a compact, interactive and standard Cobol language compiler which is ideal for the most demanding business applications. Standard features are: ISAM, Debug, ACCEPT/DISPLAY, and Interprogram Communications modules. CIS Cobol is *the* preeminent microcomputer Cobol in the industry, and the OS-9 version retains full compatibility with CP/M applications software, CIS Cobol meets the ANSI 1974 Level One COBOL standard and is GSA certified. Also available is Micro Focus' FORMS 2, an optional automatic program generator that lets you interactively design screen-oriented applications with ease.

BASIC09™ STRUCTURED BASIC INTERACTIVE COMPILER

Basic09 is the fastest and most comprehensive full Basic language available for the 6809. It combines standard Basic with the best features of Pascal. It is a unique interactive compiler that combines compiler speed, interpreter friendliness, and superlative debugging facilities. RunB, a ROMable run-time system for compiled Basic09 programs is now available as an option.

C LANGUAGE COMPILER

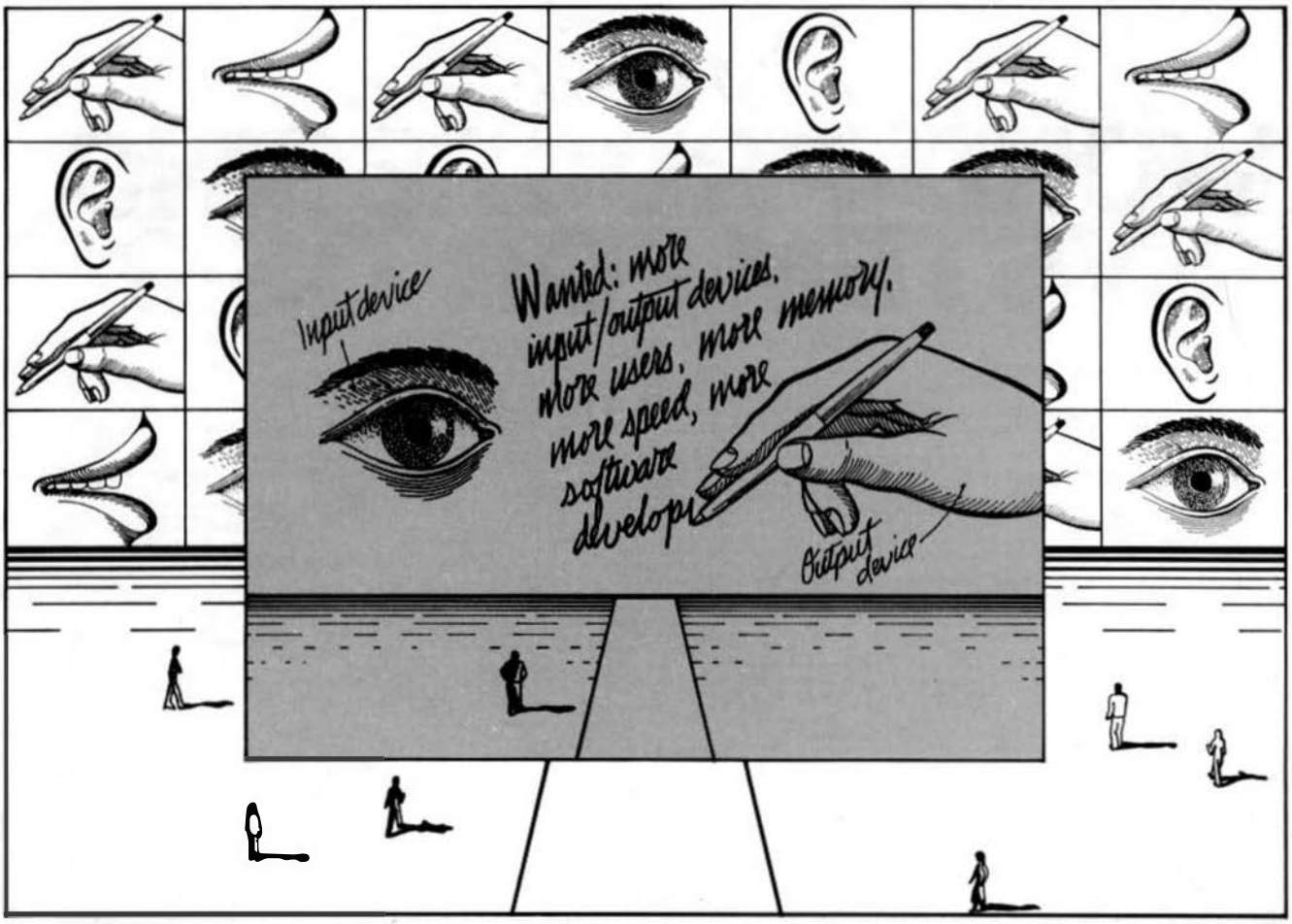
C—the systems language of the future—is here today on OS-9. This is a complete implementation of the Unix Version 7 C language including INT, CHAR, SIGNED, UNSIGNED, FLOAT and LONG data types, structures, unions, standard C library, and a full preprocessor with macro definitions. Generates fully reentrant 6809 assembly language source code output.

For information contact your computer supplier, or

MICROWARE.

Microware Systems Corporation
5835 Grand Ave., Des Moines, Ia. 50312
Ph. 515-279-8844 • Telex 910-520-2535

*Unix is a trademark of Bell Laboratories. CIS Cobol is a trademark of Micro Focus, Inc. OS-9 and Basic09 are trademarks of Microware and Motorola, Inc.



Expand your horizons with OS-9™ Level Two

Expand your 6809 computer to a fast, efficient multi-user system utilizing up to one megabyte of memory, almost any I/O device, and comprehensive implementations of the most-wanted programming languages: Basic, C, Pascal, Cobol, and Assembler.

With OS-9 Level Two, your computer is transformed into the ultimate software development system with performance and features found only on large and costly computing systems. It brings to your fingertips the friendliness and power of a Unix*-based environment. You can even run most Unix* software tools using OS-9's Unix Version 7 compatible C Compiler.

As a multiuser system, OS-9 Level Two excels with a multi-level directory system, a fast random access file system with record lockout, user name/password log-on protection, "pipes" for interprogram communication, and full file security. The versatile "Shell" command interpreter makes it easy for each user to run interactive or multiple background programs with I/O redirection to or from any file or I/O device.

As a real-time system, your OS-9-based computer can sense, monitor, control, and communicate with the real world thanks to OS-9's highly modular and user expandable structure. Adding customized I/O to OS-9 is almost too easy: software interfaces are simple, modular, and well documented. That is why OS-9 users routinely interface it to almost every kind of peripheral device and instrumentation.

For large systems, OS-9 Level Two can handle over one megabyte of RAM and ROM memory with extraordinary efficiency. When, as is often the case, two or more tasks run the same program (such as Basic09) they automatically "share" just one copy in memory. Also, OS-9 Level Two typically resides in less than 24K memory. Savings like these give OS-9 based systems large capacity without having to resort to performance-robbing techniques such as disk swapping.

OS-9 Level Two is available exclusively from manufacturers of most popular 6809 computers equipped with memory management hardware. They offer versions specifically tailored to their computers for use with both new and existing systems.

For more information about OS-9 Level Two contact your computer supplier, or

MICROWARE.

Microware Systems Corporation
5835 Grand Avenue, Des Moines, Iowa 50312
Phone 515-279-8844 • Telex 910-520-2535

*Unix is a trademark of Bell Laboratories. OS-9 is a trademark of Microware and Motorola, Inc.

DynaStar
WORD PROCESSING SYSTEM FOR OS-9

OS-9 USERS:

If your computer has a SCREEN and you're still struggling with an editor that only knows about LINES, then obviously YOU don't know about

DynaStar

DynaStar is a powerful, menu-driven, screed editor equally suited to the tasks of program preparation and document processing. With the addition of the optional DynaForm print formatter, it is the best word processing package you can buy for your OS-9 system.

DynaStar Version II is now available and features no nonsense "what you see is what you get" editing for virtually any terminal with or without cursor addressing (it must be at least able to go to "home"). To edit, simply place the cursor where you want it, and type. Any printable character you type is entered directly into your text, and any non-printable control character causes immediate execution of an editing command. Single keystroke commands permit movement of the cursor in any direction, by character, tab, word, line, or screen full, and deletion of characters, words (left or right) or a whole line. Two keystroke commands augment this set by moving the cursor to the left margin, top or bottom of the screen, beginning or end of the edit buffer, or the beginning of the next paragraph. You can search for any string, replace with any other, do it again, mark original blocks of text, copy, move or delete blocks, read or write to side-files, set tabs and margins, or center the current line.

DynaStar features automatic word wrap, and it can right-justify text as you enter it so you will see exactly how it will look before you print it. If you later make alterations or change the margins, you can reform the text a paragraph at a time with two keystrokes. For programmers, there is a special automatic indent mode to help you write well-structured code. DynaStar includes a Shell command which lets you do almost anything (including edit another file) without even losing your place in your current document, and it permits editing of large disk files in stages without forcing you to break up your files.

If you want to define more powerful commands, DynaStar includes a macro facility which lets you convert any control character to one or a string of characters of your choice. You can use this feature to create global search and replace commands, insert "boiler-plate," or simply remap your keyboard. You can also provide a special "start-up string" which is automatically executed whenever you enter the editor to set up modes such as auto-justify, display a directory, define your favorite macros, or re-map the keyboard.

For complete word-processing, we offer our DynaForm text formatter which provides all the standard features such as pagination, headers and footers with page numbers, single space, double space, multiple space, **bold face**, **double-strike**, and underline. DynaForm has its own macro facility with string variables, nested include files, a full merge-print capability for generating form letters and mailing lists, and it can generate an index automatically, sorted alphabetically or by page number. You can call it from DynaStar to proof-print the active edit buffer, or by itself to print a disk file while you edit another.

DynaStar costs only a little more than that line-oriented editor and it is available today. If you're still not convinced that it would be the best thing that ever happened to your video terminal, you can order our "Doubting Thomas" test pak consisting of complete documentation and a special version of DynaStar that lets you edit to your heart's content, but won't update your files. Later when your doubts melt away, you can obtain credit on the full purchase price and join the faithful who bought the whole thing in the first place.

"Doubting Thomas" test pak:	$ 49.95
DynaStar II (for the faithful):	$149.95
DynaForm text formatter:	$149.95
Both purchased together:	$275.00

Note: DynaStar Version I (no macros) will be available at the original price until May 31, and current owners may upgrade to Version II with full credit until June 30.

AVAILABLE SOON FOR FLEX 9

Spelltest
From Dale Puckett

FOR OS-9 AND FLEX

SPELLTEST is the most versatile 68XX spelling checker available.

MENUS MAKE OPERATION EASY. From the menu you may: Print a list of suspect words. Print a list of valid words. Check each suspect word one by one. Read your text, stopping to check suspect words. Use additional dictionaries for more thorough checking or special applications. Build an additional dictionary of newly accepted words. Write correct text file to disk.

While checking you may: Accept the suspect word. Accept and save in the dictionary. Replace with correct spelling.

Designed to be used by the layman, SPELLTEST is right at home in the office. Ease of use and speed will recover the cost in days.

22,000 word dictionary covers the first 25,000 entries in the American Heritage listing of the most common English words.

500 built-in common words (and, or, the, etc.) and 300 specific to your field, filter the text and allows a large file to processed even in small computers.

PRICE $199.00

A/BASIC
Basic Compiler
For OS-9 and FLEX

If you are still programming in assembler, this is the program for you! This BASIC compiler generates pure, fast efficient 6809 machine code from easy to write BASIC source programs. Uses ultra-fast integer math, extended string functions, boolean operators and run-time operations. Output is ROMable and RUNS WITHOUT ANY RUN-TIME PACKAGE. Supports IF-THEN-ELSE structure, random access, and several improvements over the original 6800 version sold by Microware. Optimized for the 6809, A/BASIC is 5 to 10 times faster than the original 6800 version, and produces code approximately 30% smaller.

SPECIAL

CHESS program coded in A/BASIC (originally sold for $50) is included FREE on the disk in both source and object for your enjoyment. Also some utilities are included for testing and examples, all in source on the disk!

ONLY $150.00

specify OS-9 or FLEX

TOOLKIT NO1

The Basic Programmers *Toolkit*
by Dick Bartholomew

The Basic Programmers Toolkit gives the BASIC programmer the power and flexibility never before achieved under FLEX.

PRICE $49.95 object only
$69.95 with source on disk!

TOOLKIT NO2

The Programmers *Toolkit*
by Dick Bartholomew

The Programmers Toolkit is a package of utilities and programs that extend the capabilities of FLEX to the utmost.

PRICE $49.95 object only
$69.95 with source on disk!

CRASMB

MULTI CPU CROSS ASSEMBLER FOR 6809
FLEX
by Frank Hoffman

CRASMB is a conditional macro assembler with the capability to use different CPU overlays in order to cross assemble. These CPU overlays called *CPU PERSONALITY MODULES (CPM's)* can be called from a source file, thereby making it easy to create object code for a variety of CPU's. It is also possible to create new *CPM's* yourself for any 8 or 16 bit CPU. The information needed is included in the manual. If you decide to do this, it would be advisable to purchase the source for one of the *CPM's* and modify it rather than starting from scratch. *CPM's* are currently available for the following CPU's: 6809, 6800, 6805, 6502, Z80/8080, 1802, and others coming.

PRICE $139.95

Includes one 8 bit *CPM* of your choice (not source)
Additional *CPM's*
8 Bit $25.00 *Source $25.00 extra*

PLOT

Now you can have GRAPHICS added to all your programs. Just write the data out to a virtual array and call PLOT. PLOT is written in TSC XBASIC and the source is included on the disk.

INFINITE RESOLUTION GRAPHICS ON YOUR TERMINAL OR PRINTER. HISTOGRAMS, BARGRAPHS, XY PLOTS PLUS OTHERS.
IN TSC XBASIC. SOURCE INCLUDED ON DISK. $44.95

Dynasoft PASCAL 1.4 for OS-9

Dynasoft Pascal 1.4 includes all the features of the FLEX version 1.3 with the following enhancements: Chain, Fread, Fwrite, Seek, Open, Create, Close, Delete, Fork, Send, Wait, Sleep, Setime, Time, Getstelus, Setstatus, SetPriority, GetProcID, and JSR. This is an excellent and fast program, small enough to write utilities but powerful enough for things like DynaStar.

Object only $69.95
Add for run-time source on disk $30.00
Add for source of Dynasoft Pascal itself $125.00

THE BILL PAYER SYSTEM™

THE BILL PAYER is a package of 10 menu driven programs in TSC Extended Basic. This powerful system helps you keep track of your bills. You can create a vendor list, enter invoices to be paid, generate reports about them, print your checks and much more. Uses random access files.

Explore Package now included at the same price.

THE PURCHASE ORDER system adds purchase orders to the BILL PAYER. This package of programs adds another level of control to your expenditures. Prints out purchase orders and keeps track of purchases. Requires the Bill Payer to work.

INCOME/EXPENSE LEDGER. This valuable package is most appreciated at tax time. Allows up to 99 income and expense numbers. Ties into the PURCHASE ORDER system, and the Bill Payer.

Includes manual and source supplied on disk in TSC Extended Basic.

THE BILL PAYER
PURCHASE ORDER
INC/EXP LEDGER

ALL FOR $169.95

COLOR COMPUTER USERS

THE POWERFUL FLEX DISK OPERATING SYSTEM WITH HUNDREDS OF SOFTWARE PACKAGES IS NOW AVAILABLE!

Now you can run FLEX, OS-9 and Radio Shack disk software on your Color Computer. If you have a 32K Color Computer with the Radio Shack disk system, all you need to do is make a trivial modification to access the hidden 32K, as described in the Feb. issue of COLOR COMPUTER NEWS and the April issue of '68' Micro. You can get FLEX from us right now. OS-9 will be ready by summer. Please note that this will only work with the Radio Shack disk system and 32K/64K memory chips that RS calls 32K. Maybe they put 64K's in yours, too. If you don't have a copy of the article, send a legal size SASE (40¢ stamps) and we'll send it to you.

Using this system to run FLEX and OS-9 has many advantages. First, it gives you 48K from zero right up to FLEX. This means that ALL FLEX compatible software will run with NO MODIFICATIONS and NO PATCHES! There are no memory conflicts because we moved the screen up above FLEX which leaves the lower 48K free for user programs.

What you end up with is 48K for user programs, 8K for FLEX and another 8K above FLEX for the screens and stuff. We have a multi screen format so you can page backward to see what scrolled by and a Hi Res screen that will enable us to have 24 lines by 42 character display is on the way. That's better than an Apple!

We also implemented a full function keyboard, with a control key and escape key. All ASCII codes can now be generated from the Color Computer keyboard!

We also added some bells and whistles to Radio Shack's Disk system when you're running FLEX or OS-9. We are supporting single or double sided, single or double density, 35, 40 and 80 track drives. If you use double sided drives, the maximum is three drives because we use the drive 3 select for side select. When you are running the Radio Shack disk, it will work with the double sided drives but it will only use one side and only 35 tracks. Using 80 track drives is okay, but will not be compatible with standard Radio Shack software. You can also select each drive's stepping rate and drive type (SSot, SD or DD).

In case you don't understand how this works, I'll give you a brief explanation. The Color Computer was designed so that the roms in the system could be turned off under software control. In a normal Color Computer this would only make it go away. However, if you put a program in memory to do something first (like boot in FLEX or OS-9), when you turn off the roms, you will have a full 64K RAM System with which to run your program.

Now, we need the other half of the 64K ram chips to work, and this seems to be the case most of the time, as the article states. Of course, you could also put 64K chips in.

Some neat utilities are included.

MOVEROM moves a Color Basic from ROM to RAM. Because it's moved to RAM you can not only access it from FLEX, you can run it and even change it!! You can load Color Computer cassette software and save it to FLEX disk. Single Drive Copy, Format and Setup commands plus an online help system are included.

Installing FLEX is simple. Insert the disk and type:

RUN "FLEX"

That's all there is to it! You are now up and running in the most popular disk operating system for the 6809. There are hundreds of software packages now running under the FLEX system. Open your Color Computer to a whole new world of software with FLEX.

FLEX $99.00
INCLUDES OVER 25 UTILITIES!

Other languages available include; FORTH, Pascal, Fortran77, 'C,' A/BASIC compiler, plus more.
Application packages include; A/R, G/L, A/P, Inventory, Electronic Spreadsheets, Accounting, Database programs and more. SEND FOR LIST.

TRS-80 COLOR COMPUTER COMPLETE WITH 64K RAM, 24K ROM, SINGLE DISK DRIVE AND FLEX, SET UP AND READY TO RUN FOR ONLY $1,275. Includes 60 day extended warranty. If you have a Computer, call about RS disk controllers and drives.

SOFTWARE CATALOG

PROGRAM	OBJECT ONLY/SOURCE		code
BILLPAYER		169.95	x
PLOT		44.95	x
TABULA RASA		100.00	x
Mailing List		99.95	x
Forms Display		49.95	x
Inventory with Material Requisition Planning		100.00	x
Some Common BASIC Programs		69.95	x
FLEX For Color Computer	99.00		9
X-FORTH (FLEX)	149.95		8 & 9
CC-FORTH (TRS-80 Color)	99.95		9
TOOLKIT #1 (BASIC)	49.95/	69.95	9
TOOLKIT #2	49.95/	69.95	9
AUTOTASK		129.95	9
A/BASIC Compiler	150.00		9
Extended Utilities	49.95/	69.95	9
Password Protection	69.95/	89.95	9
CRASM8 (X Assembler)	139.95		9
Personality Modules (1 INC) 6502, 6800, 6805, 6809, Z80, 8080, 1802	25.00	50.00	9 ea
READTAPE		54.95	9
SPELLTEST	199.00/	299.00	9
READTEST	54.95/	74.95	8 & 9
ESTHER	39.95/	59.95	8 & 9
HELP	29.95/	49.95	8 & 9
Job Control Program	49.95/	89.95	8 & 9
DYNASOFT PASCAL (FLEX)	59.95/	89.95	9
DYNASFT PASCAL (OS-9)	69.95/	99.95	9
DYNASOFT Compiler Source		125.00	P
DYNASTAR Screen Editor (OS-9)	149.95		9
SUPER SLEUTH (6800/6809)		99.00	8 & 9
SUPER SLEUTH (Z80)		99.00	8 & 9
CROSS Assembler Macros for TSC ASMB 6800/1, 6805, 6502, Z80, 8080/5	3 FOR	49.95 99.95	each
6502 Translator		75.00	9
Debugging Simulators 6805 or 6502		75.00	9 ea
STYLOGRAPH 4.2 D	295.00		9
STYLOGRAPH MAIL MERGE	125.00		9
STYLOGRAPH Spelling Checker	145.00		9

CODE X = XBASIC, 9 = 6809, 8 = 6800, P = PASCAL

Software by Technical Systems Consultants, Inc.

Flex™ (includes Editor & Assembler)	150.00
UniFLEX™ (includes one year maintenance and update)	450.00
Editor	50.00
Assembler	50.00
68000 Cross Assembler on 6809	250.00
Text Processor	75.00
Extended Basic	100.00
Basic Precompiler (specify standard or extended)	50.00
Pascal (Flex™)	200.00
Pascal (UniFLEX tM) (Add $75.00 for one year's maintenance and update)	225.00
SoftMerge Package	75.00
6809 Flex™ Utilities	75.00
Debug Package	75.00
Diagnostic Package	75.00

Software by Microware Systems Corp.

OS-9™ Level One Operating System	200.00
OS-9™ Level Two Operating System	500.00
BASIC09™	200.00
OS-9™ Macro Text Editor	125.00
OS-9™ Interactive Assembler	12500
OS-9™ Interactive Debugger (Disk version)	50.00
CIS Cobol Compiler	895.00
Pascal Compiler	400.00

USA add $2.50 for standard UPS shipping & handling
Foreign orders add 20% Airmail
Specify 5" or 8" disk and 6800 or 6809
VISA—MASTER CHARGE—DINERS CLUB ACCEPTED

OUR SOFTWARE IS GIMIX COMPATIBLE

AUTOTASK

WITH MENU

AUTOTASK with MENU is a revolutionary new concept designed to overcome the problems and frustrations which confront the non-technical when using a computer. Users are greeted with a series of self-prompting interactive menus linking directly to the application. Several example menus are provided. You can create your own menus from simple text files. AUTOTASK with MENU gives you unlimited software flexibility by providing a system to coordinate multiple-application programs.

Bundle several different software packages to present a coordinated system to the user. AUTOTASK with MENU is compatible with all FLEX compatible software. It uses very little memory and is easy to learn.

PRICE $129.95
Includes source on disk!
Manual $10.00

6502 TRANSLATOR
Translator 6502 code to 6809
$75.00

INVENTORY
with MATERIAL
REQUISITION PLANNING
$100.00

SUPER SLEUTH
Disassembler for 6800/6809 or Z80
$99.00

TABULA RASA
Electronic Spreadsheet
$100.00

TRS-80 COLOR COMPUTER

FORTH FOR THE TRS-80 COLOR COMPUTER DISK SYSTEM

Trying to get control of your Color Computer?? Tired of translating HEX to decimal?? Tired of remembering where the VDG and SAM are and how to program them?? Want to write machine language code with assembly language mnemonics instead of POKES?? Want to write programs in half the time?? Want to write lots of small pieces of code that you can put together in seconds to do BIG JOBS??? Want a language that is at least 5 to 10 times faster than BASIC?? Want to learn everything there is to know about FORTH, with the best manual on the market, including lots of examples of FORTH applications, and detailed explanations of how everything works??

CC FORTH IS THE ANSWER!!
Includes Editor, 6809 Assembler
String Functions, Disk Data File
Operations and Much Much More!

$99.95

FORTH

FLEX COMPATIBLE FORTH

BY Chuck Eaker, Ph.D.
X-FORTH NOTES
Supplied on one 8" disk or 2 5" disks,
with a 400+ page manual.

Disk(s) have the source of everything but the core.
PRICE only $149.95 plus $2.50 S&H
Manual available seperately for $49.95 plus $2.50 S&H

HARDWARE CATALOG

CRT TERMINALS

Terminal		Cash	Credit Card
TVI	910	588.50	605.00
TVI	912/C	709.00	728.00
TVI	920/C	758.00	779.00
TVI	925	759.00	780.00
TVI	950	976.00	1004.00
Zenith	Z19	700.00	719.00
SOROC	IQ130	589.00	605.00
SOROC	IQ135	765.00	78 .00
SOROC	IQ135G	8 1.00	853.00

PRINTERS

Printer		Cash	Credit Card
IDS	445	632.50	650.00
IDS	460G	852.50	876.00
IDS	560G	1094.50	1124.50
Centronics	739	519.00	533.50
Centronics	739-3	641.50	658.79
TI	820	1738.00	1785.50

All printers and terminals are shipped freight collect and there are no COD's

FRANK HOGG LABORATORY
130 MIDTOWN PLAZA
SYRACUSE, NY 13210 (315) 474-7856

EPSON DOT MATRIX PRINTERS

The Epson MX-80

The Epson MX-100

REDUCED PRICE SPECIALS

80-Column

~~$495.00~~ $455.00

MX-70 ~~$395.00~~ $355.00

136-Column

~~$798.00~~ $725.00

MX-80 FT ~~$625.00~~ $575.00

FACTORY FRESH — LATEST PRODUCTION — BRAND NEW

SWTPC-Motorola, MP32 Dynamic Memory Board **ONLY $199.00**

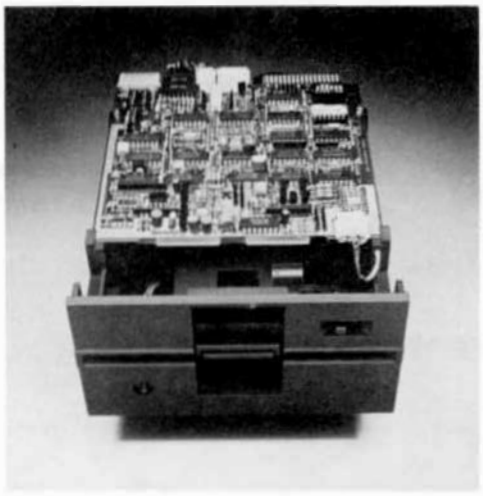
Assembled & Tested

Thousands sold at $650.00 Save over $450.00

1 MHZ - No extended addressing

Can be set up for $0-7FFF or 8000-FFFF

Limited Quantity (less than 50)



SOUTH EAST MEDIA

P.O. Box 794 Chattanooga TN 37443

1-615-842-4601

** DISK DRIVES & CABINETS W/PS **

5 1/4"

TEAC - Single Sided, Double Density, 40 Track.....$249.95

TEAC - Double Sided, Double Density, 40 Track.....$349.95

CABINET - Single Drive with Power Supply.....$ 79.95

CABINET - Double Drive with Power Supply.....$ 99.95

CABLE - Single Drive.....$ 24.95

CABLE - Double Drive.....$ 34.95

* NOTE - When ordering cables please specify S50 Bus or Other!!!

Call or write for disk controller Board information.

SHIPPING: - Memory Boards or Cables, Add $4.50; Printers, Add $7.50
- Disk Drives or Cabinets with Power Supply, Add $7.50

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

C DUMP PROGRAM

As I mentioned last month, this month's column will be a discussion of the Memory Dump program presented in Pascal last time, converted to C. Since I've received some valuable input from Norm Commo, author of the column on C, I will include his version of the program too.

Have you just gotten used to the use of BEGIN and END in Pascal, and found it rather nice to enclose comments in "curly braces" ? You are in for a jolt. In C, BEGIN is replaced with {, and END with }. The end of a statement is indicated with a semicolon just as in Pascal. C has eliminated the use of the words DO and THEN, and generally reduced the wordiness of Pascal. Variable declarations are present but different than in Pascal in that the type of the variable comes first, and there is no heading as such for the VAR section as in Pascal. The alternate comment delimiters in Pascal are, as you may remember, the double character combinations (* and *). In C, one uses /* and */. These are not quite as handy because the "foreslash" or fraction bar is typed in the lower case mode, while the asterisk is upper case. This necessitates use of the shift key at each end of a comment, and in reverse order, and I, at least, always get out of sync at one end or the other. On my terminal this produces the delimiters ?: and :?, which C doesn't like at all.

Having a copy of Duggers small C for evaluation, and having just received an update of the original, I decided to strike out on my own and see if I could translate my Pascal program into C. This is not always the best approach because translations are not necessarily the most efficient. Sometimes you have to think in the language from scratch in order to arrive at the best solution. Anyway, I did translate the program. My skill level with C might be revealed by the fact that at one point, I had over 30 compiler flagged errors in the two page program. After getting rid of those, it still didn't run for beans. Turned out that I had overlooked or forgotten a detail. Strings in C are delimited by double quotes, but character constants and values are delimited by single quotes. When I got that straightened out, I was a long way toward having the program running.

You might note that the main difference in organization between C and Pascal is that all the subroutines (functions in C, and FUNCTION or PROCEDURE in Pascal) are in a different place. Pascal requires them to be ahead of the main program, and C requires them to follow it. In C, functions are distinguished from variable names by the fact that they have parameters. Therefore you must include the parentheses that surround the parameter list even if there are no parameters passed to the function. I don't like typing printf ("\n"); when I want a linefeed and CR, so I wrote the function skip(). Skip simply outputs a CR and an LF in that order. No parameter is sent to it, and none is returned. You may see that it would be easy to pass a count to it to output multiple linefeeds, but I only needed one in all cases, so I didn't bother.

It is a feature of C that arithmetic overflow is not checked, so that an integer may just as well represent an address (an unsigned integer value), as a signed integer. That simplifies the handling of page, making the treatment the same as in the OmegaSoft Pascal where HEX type is available. This C has only one loop control method implemented, the while loop. I had used a FOR NEXT loop in Pascal, and a REPEAT UNTIL. The conversion presents minor problems. In the case of converting a FOR NEXT to a WHILE, it is only necessary to initialize an index variable to 0 before the loop, and increment it at the end of the loop. The while checks for it passing the upper limit of the loop and the loop terminates nicely. A REPEAT UNTIL may be duplicated, the only consideration being that the variable to be tested must be initialized to some value that is included in the while condition or the loop won't be executed at all.

I used the library function putchar for most of the output. Putchar allows output of one or two characters at a time. You may note seemingly careless mixing of integer and character types of data. For example, in the function hex, you will see a statement that reads "ch = number - 10 + 'A'; ch is a character variable, number an integer, 10 is obviously an integer constant, and 'A' a character constant. C is designed to allow this. All characters are converted to integer, ie 2 bytes whenever they are used in a statement. They are stored as single bytes. The programmer must be careful not to assign a value larger than a single byte to a character. This implicit conversion of types makes the program simpler and requires much more care on the part of the programmer if the program is to be successful. Another example of this is in the skip function where the LF and CR are output simply with putchar(13) and putchar(10).

I have used the same variable names in the C version as the Pascal versions, so any difference you see in the programs is due to the language and not the use of long variable names in one and short ones in the other. One difference is that all keywords in C MUST be lower case. Tradition is to use lower case for variable names as well, and all upper case for constants, one of which I have used in this program. The program in C (final binary) is $A20 bytes long, just a little shorter than the OmegaSoft version. Execution seems faster than OmegaSoft, though I didn't make any accurate timing tests. C buffers all input through the library function getchar(), and I didn't like having to hit return to enter commands, so I implemented getch() as a jump to FLEX GETCHR. I determined that parameters are passed to a function (in this implementation) on the 6809 user stack, and any returned value is passed in the X register. The main program cleans up the stack on return from the function. That is, the function doesn't have to remove the parameters from the stack, just read them.

Obviously, C allows in line assembler code by means of the compiler directives #asm and #endasm which must start in the first column of a line. All text between those two symbols is passed straight through the compiler as it generates assembler source code for everything else in the text file. You may see a very strong resemblance between the getbyte function in C and the GETBYT code in the assembler external for the OmegaSoft version. The differences are due solely to the difference in the way parameters are handled in the two compilers. You may have noted that the assignment statements in C use = rather than the := of Pascal, and the test for equality is == in C rather than the = in Pascal. Just a little confusing at first, though more efficient, since (usually) there are more assignment statements in a program than equality tests. C uses != for not equal, and % for MOD. One other feature of C is shown in the listing. #define CR 13 defines the identifier CR as being replaced with the integer value 13 wherever it appears in the program. This is exactly like an equate in assembler. It is equivalent to using CR EQU 13 (or $0D) at the start of the assembler code. Wherever CR appears in the C source, 13 is substituted in the assembler source output of the compiler.

The last thing that needs mention, is the increment feature. In the code you will see the statement line++; That means to increment the variable line by 1 (NOT 2). It is necessary to use consecutive plus signs to distinguish this operation from an addition.

The only thing that disturbs me a bit about C is that the same symbol is used for different purposes in different contexts in the code. For example, the symbol '&' is used between two boolean expressions to represent the AND function of those two expressions. It is also used before a variable name to indicate a pointer or the address of that variable &line, for example would represent the address of the variable line, not its contents. Another example is the percent sign which means MOD in an arithmetic calculation, and is used to signal a format description when used in a string in connection with the library function printf. printf ("%d",number) for example would print the value of the variable number as a decimal number. Since it must make the design of the compiler more difficult to have to distinguish context, I wonder why the designers of C did it that way?

Now, on to Norm Commo's program. Norm was very kind in the comments at the beginning of his program, and I really appreciate his willingness to participate in my column effort, since he has his own to look after too. First I must comment that Norm's use of #define to define constants for the program make it better than my effort in two ways. First of all, PAGSIZ indicates to the reader of the program what is going on much better than the literal 256. Secondly, this program could be adapted to other applications where it might be desirable to show, for example 128 bytes on the screen. I can think of some systems that don't display long enough lines on their CRT to allow 16 bytes in Hex and ASCII with the spaces between them. The format generated by the program here, is 72 columns wide. It would be useless on a 64 character terminal. With the constants, it is a simple matter to adjust the "parameters" to make the output fit your display.

Notice Norm's interesting way of making comments stand out, by using asterisks as is normally done in assembler programming. The first place where Norm's code is significantly more efficient than my effort is in the two lines: line = -1; while (++line < LINS PER PAGE). Norm has included the incrementing of the loop index variable in the test (++line). Of course incrementing the index at the beginning of the loop requires it to be initialized to one less than the first desired index, hence the previous line is line = -1;. Norm again used this technique in the next two while loops. He has also made beter use of functions, having written a function to output four hex digits, and one to output two hex digits.

The variable mem is defined as integer, making it 16 bits long. Norm indicates that he did that because of a small problem with the character handling in Duggers C. Probably the one most significant feature of C that I completely missed (or should I say didn't understand at all) is the pointer feature. The use of the pointer and of the logical shift are shown in the line that reads: mem = * (pageadd + (line * BYTS PER LIN) + bytcnt) >> 8; What's that '*' doing in fronf of the expression? We BASIC and Pascal users thought '*' meant to multiply! In this context in C, the '*' means "POINTER". That is, the calculated value of the expression is a pointer to the place in memory where the value of mem is to be read. Since mem is an integer, the pointer will point to the high order byte. Norm has used >> 8 to right shift this value 8 places, making the high order byte 0 and the low order byte of the result the value in the desired memory location. I must admit that I hadn't picked up the shift operation either. The loop that includes this line (the first time) outputs the 16 hex values in a line with spaces between.

The next while loop gets the same 16 memory location's contents, and outputs them as characters. Here a slight philosophical difference between my program and Norm's appears. He ignores any "character" with a value greater than $7F and prints a period, which we both do for control characters (less than $20). I've found that one major use of the DUMP program is to scan memory for ASCII text, words or messages. Some programs set the high order bit of ASCII characters as a flag for the program, for example, the first letter of a label, name, etc. If you simply ignore any value with this bit set, you will not make sense of the output. My program therefore throws away the high order bit (which is normally the parity bit for ASCII codes) and prints the character that results from ignoring that bit. You may take your choice.

The program next gets a command from the user. The use of the library function 'tolower' is a good idea. It converts upper case letters to the lower case, so that it doesn't matter if you enter 'F' or 'f', you still go forward by one page. Other than that added touch, the lines that get the command and perform the necessary operation are equivalent to my approach.

We now come to the functions. 'buff' is an array of characters of size BUFSIZ. The library function 'gets' is passed the name and size of the array, and it gets the characters input from the termnal. Again, a library function 'isdigit' as been used to test the character for being a digit. Rather than my if (ch >= '0' or ch <='9'), Norm has used if (isdigit (ch)). Here, however, I ran into a problem. Norm had mentioned to me that he had customized the Dugger runtime package a bit. My version doesn't recognize the function 'gets' which I assume Norm added to get a string. I've commented Norm's original line out, and used the scanf function of Dugger's library. Norm will notice that I have rearranged the code after the comment /* convert to hex */. Apparently Norm has improved the tolower function in Dugger's library. Norm had used ch = tolower (buff[i++]); and then tested with isdigit. I found that the tolower function doesn't check first to see if it is given a valid upper case letter. It tried to convert the ascii digits 0 through 9 to lower case, and in the process made them unrecognizable. (They were converted to the upper case letters P through Y). I simply get the character and test it with isdigit first, and then if it is not a digit, use tolower and proceed with the conversion. I wanted to make this program completely compatible with the Dugger runtime so that we could compare two completely equivalent programs that you readers who have Dugger's C can run. I have noted one other peculiarity with Dugger's runtime function getchar. I get a double character on my terminal. The FLEX GETCHR routine (which is called by the Dugger runtime package in the getchar code) echoes the character, and getchar does too.

At this print, Norm's code follows my assembler version more than the Pascal program. He has defined a couple of functions, put_2_hex (num) and put_4_hex (num) which do essentially what the SBUG routines OUT2HS and OUT4HS do.

The function puthex (num, nybble) is the most interesting part of the program, because it is a recursive routine to output the number num, as nybble hex digits. That is, the number and the number of digits to be output are passed to the function. It calls itself recursively in the statement that reads if (nybble > 1) puthex ((num >> 4),nybble - 1); This line causes puthex to be called recursively, each time with nybble reduced by one, until it reaches zero. At that point, the values of num after successive shifts right four times, are all stored in local variables on the stack. They are then successively output by the remainder of the code, which is similar to mine. I had not realized that the AND function (&) is a bitwise AND, or I might have used it for a mask. The AND in Pascal is generally only used for BOOLEAN evaluation, though some implementations allow its use on a bitwise basis too. The statement num >> 4 means to shift num to the right by four places. num in this case is an integer variable 16 bits long. A left shift is also valid. These instructions are very fast ways to multiply or divide an integer number by an even power of two.

As you can see by the finished programs, C allows more flexibility than Pascal and is slightly less wordy. The code generated by the Duggers compiler was slightly less than the code generated by the most efficient Pascal compiler, but Duggers C always compiles the whole runtime package. Which implies that perhaps the C would show a larger advantage over Pascal when used on a larger program, because the code generated per line of user source code may be less.

MORE COMING ON ASSEMBLER PROGRAMMING

As this is being concluded, I have received the first feedback from my February column, in which I asked for some indication of whether you readers want more on Assembler programming and/or Lucidata's implementation of Pascal. So far, those writing have indicated most strongly, an interest in the Assembler programming techniques, a lesser but reasonably strong interest in the Lucidata discussions, and some interest in exploring C compilers a bit. Several people have expressed interest in my 6809 Arithmetic package too. There were several contributions of information for the Assembler programming series, and we will hit that area hard next month.

```
/* HEX ASCII DUMP PROGRAM IN C */
#define CR 13
#define LF 10

int k, l, m, n, line, page;
char ch;

main ()
{
   page = getpage();
   ch = 'X';
   while (ch != 'E')
   {
      skip();
      n = page * 256;
      line = 0;
      while (line <=15)
      {
         k = 16 * line;
         putchar (hex (page / 16));
         putchar (hex (page % 16));
         putchar (hex (line));
         putchar ('0');
         putchar ('  ');  /* two spaces */
         l = 0;
         while (l <= 15 )
         {
            m = n+k+l; /* aritheetic overflow OK here */
            m = getbyte (m);    /* assign char value to integer OK */
            putchar ( hex (m / 16));
            putchar ( hex (m % 16));
            putchar (' ');
            l++;
         }
         putchar (' ');
         l = 0;
         while (l <= 15)
         {
            m = n+k+l;
            m = getbyte (m);
            m = m % 128;
            if (m > 31) putchar (m);
            else putchar ('.');
            l++;
         }
         skip();
         line++;
      }
      skip();
      printf ("command? ");
      ch = getch ();

      if (ch == 'F') page = page + 1;
      if (ch == 'B') page = page - 1;
      if (ch == 'N') page = getpage();
   }
   exit();
}

skip ()
{
   putchar (CR);
   putchar (LF);
}

getbyte (add)
   int add;
{
#asm
 LDB [0,U]  ADDRESS PASSED ON USER STACK
 CLRA
 TFR D,X    CHAR TREATED LIKE INTEGER WHEN RETURNED IN X
 RTS
#endasm
}

getpage ()
{
   int page;
   char ch;
   int k,n;

   printf (" starting page (two hex digits)? ");
   n = 0;
   k = 1;
   while (k <= 2 )
   {
      n = n * 16;
      ch  = getch ();
      if (( ch  >='0' ) & ( ch  <='9'))
           n = n + ch  - '0';
      if (( ch  >= 'A') & ( ch  <='F'))
           n = n + ch  + 10 - 'A';
      k++;
   }
   return n;
}

hex (nuober)

   int nuober;
{
```

```
   char ch;

   if ((number >= 0) & (number <=9)) ch = number + '0';
   if ((number >=10) & (number <=15)) ch = number -10 +'A';
   return ch;
}
getch ()  /* unbuffered get character routine in FLEX */

{
#asm
   JSR $CD15
   TFR A,B
   CLRA
   TFR D,X
   RTS
#endasm
}

/*
 * memory.ccc    rev:  1
 * n f comma
 *
 * created:      2/1/82
 * last edit:    2/2/82
 *
 * A program to read what's in memory.  Originally
 * written by Ron Anderson.  Updated by More Comma
 * as a comparison between a program written by a
 * neophyte C programmer versus a reasonably facile
 * C programmer.
 */

#define PAGSIZ 256          /* the size of a page */
#define BUFSIZ 10           /* a tiny line buffer */
#define NYBMASK 15          /* a four bit mask    */
#define LINS_PER_PAGE 16
#define BYTS_PER_LIN 16
#define RUBOUT 128
#define CR 13
#define LF 10

main ()
   {
   char ch;
   int bytcnt, line, pageadd, mem;

   pageadd = getpage();
   ch = 'X';
   while (ch != 'e')
      {
      skip();
      /* print a page of lines */
      line = -1;
      while (++line < LINS_PER_PAGE)
         {
         /* print the line address */
         put_4_hex (pageadd + (line * BYTS_PER_LIN));

         /* output hex bytes */
         bytcnt = -1;
         while (++bytcnt < BYTS_PER_LIN)
            {
            mem = * (pageadd + (line * BYTS_PER_LINE) + bytcnt) >> 8;
            put_2_hex (mem);
            }
         putchar (' ');

         /* output ascii characters */
         bytcnt = -1;
         while (++bytcnt < BYTS_PER_LIN)
            {
            mem = * (pageadd + (line * BYTS_PER_LIN) + bytcnt) >> 8;
            if (' ' < mem & mem < RUBOUT)
               putchar (mem);
            else
               putchar ('.');
            }

         skip();
         }

      skip();
      printf ("command: ");
      ch = tolower (getchar());

      /* decode first */
      if (ch == 'f')
         pageadd = pageadd + PAGSIZ;
      if (ch == 'b')
         pageadd = pageadd - PAGSIZ;
      if (ch == 'n')
         pageadd = getpage();
      }
   }


/*
 * get the page number as two hex digits (the msb
 * of the address) and return the base address of
 * the page
 */

getpage()
   {
   char buff[BUFSIZ], ch;
   int i, n;

   skip();
   printf ("\nstarting page (two hex digits): ");
/* gets(buff,BUFSIZ);  */
   scanf ("%s",buff);

   /* convert to hex */
   n = i = 0;
   while (i < 2)
      {
      n = n * 16;
      ch = buff[i++];

      if (isdigit (ch))
         n = n + ch - '0';
      else
         ch = tolower (ch);
```

```
    if ('a' <= ch  & ch <= 'f')
       n = n + ch - 'a' + 10;
    }
  return (n & PAGSIZ);
  }

/*
 * output 2 hex digits and a space
 */

put_2_hex(num)
  int num;
  {
  puthex (num,2);
  putchar (' ');
  }

/*
 * print 4 hex digits and two spaces
 */

put_4_hex (num)
  int num;
  {
  puthex (num,4);
  putchar (' ');
  putchar (' ');
  }




 /*
  * output a hex number recursively
  */

 puthex (num, nybble)
    int num, nybble;
    {
    int digit;

    /* if not last digit, call again (recursively) */
    if (nybble > 1)
       puthex ((num >> 4),nybble - 1);

    /* print out the present rightmost nybble */
    digit = num & NYBMASK;
    if (0 <= digit & digit <= 9)
       putchar (digit + '0');
    else
       putchar (digit - 10 + 'A');
    }

 skip()
    {
    putchar (CR);
    putchar (LF);
    }
```

"C" User Notes

Norm Commo
3 Pryor Road
Natick, MA 01760

One of the subtle strong points of C is that there are no predefined functions, such as i/o, as there are in other languages. This statement may seem absurd at first glance. But consider this, if it's not defined, then you can make any function what you want it to be. How often have you said to yourself "if only it worked this way". In C you have complete freedom to insure that every function operates the way YOU want it to, not the way that some vendor envisioned most users would like it to.

Does that mean the the neophyte C user is stuck with a language that does nothing until he write the most elementary i/o functions? Not really. There exists a body of functions described in K&R(1) that are considered standard. All C compilers come with some subset of these or their functional equivalents, and every C compiler for the 6809 that I've recieved so far has included the source code for their library as part of the package.

What functions should you expect to get when you buy a C compiler? That's what we'll cover this in this month's column. Bear in mind that a vendor may not supply all of them, or supply them exactly as they are described here. The descriptions will be taken directly from "The C Programming Language" wherever applicable.

A standard format will be used to describe the functions. It will consist of a call to the function including any required parameters, a type declaration of those parameters, what is returned by the function, and finally, any action done by the function that is not implied by the previous parts.

TERMINAL I/O

Most terminal i/o in C can be handled by four basic functions. There are a few others, but I would be very surprised if these four didn't comprise 90%, or more, of your terminal i/o calls.

```
getchar()
   returns    The character from the terminal.

   action     Gets a character from the terminal.
              Depending on the vendor, there may
              be some "filtering" of what actually
              constitutes a valid character.
  gets(s, n)
     char *s;   A pointer to a buffer to be filled.
     int n;     The length of the buffer.  Some
                libraries will not use n.
     returns    The number of characters actually
                put in the buffer, excluding the
                NULL terminator.

     ection     Fills the buffer pointed to by s
                with characters from the terminal
                until <cr> is struck.  The <cr> is
                not put in the buffer, which gets
                terminated with a NULL.
                If n is not a paramter of gets() the
                buffer had  better be big enough to
                hold all the input characters, or
                something will get clobbered when
                then end of the buffer is exceeded.


 putchar(c)
    char c;    The character for output to the
               terminal.
    returns    Some versions will return c, other
               will return garbage.
```

action Outputs the character c to the terminal.

```
printf(frmt, arg1, arg2, ...)
   char *frmt;  The format string.
   arg1, arg2;  Optional arguments, as many as are
                called for in frmt.
   returns      garbage
```

Think of "printf" as mnemonic for "print formatted". For indeed, that is what the function does. The first argument is a string referred to as the format string. Print() reads the string a byte at a time, considering each byte as either a printable character, or a special action character that determines how to handle the next argument in the list.

There are two types of special characters that can appear in the format string. One is the backslash character "\", called the escape character. The other is the percentage sign "%", which preceeds a "conversion" character.

The backslash is used to denote certain nonprintable characters, or any character that normally has special meaning to the compiler, but must now be taken literally. The common escape characters are

\b	backspace
\f	formfeed
\n	newline (line feed)
\r	carriage return
\t	tab
\\	backslash
\'	single quote (for a character literal)
\"	double quote (embedded in a string)
\ddd	a character whose octal values is ddd
\0	null, a special case of \nnn

Some compilers may also include the escape sequences

\xdd	or
\0xdd	the character whose hex value is dd

Which escape sequences are available to you will depend on your compiler. Rightfully, a compiler will translate escape characters into the appropriate byte code. This will remain true whether the escape character occurs in a #define, a character literal or a string. However, some "Small C Compiler" derivatives will not. Check your compiler manual carefully. If they don't tell you enough or you're not sure, then write a little test program with the sequence in question and see what the code looks like. Both Dugger and Intersoft, in the first release of their compilers, pass escape sequences in a string right through to printf(), where they are then "interpreted" at run time.

The other special character is the percentage sign "%". It signals that a "conversion sequence" will follow terminated by a conversion character. When printf() encounters the % in a format string it takes the next argument in the list and outputs the argument as dictated by the conversion character and any other optional characters in between them.

The optional character occuring between the % and the actual conversion character define the field in which the argument will be printed and how it will fit in that field. The allowable modifiers are

A minus sign, which specifies left justification of a string or a converted number in its field.

A digit string, which specifies the minimum field width. A converted number will be at least this wide, or bigger if necessary. A number smaller than the specified field is padded on the left or right. It is padded with spaces unless the field specifier began with a zero, in which case it is zero padded.

A period, which delimits the field specifier from the next digit string.

A digit string which specifies either the maximum number of characters to be printed from a string, or the precision of a float or double.

An "l" (ell), which specifies that the argument is to be taken as a long rather than an int.

The conversion characters and their meanings defined in the standard are

- c the argument is printed as a single character
- s the argument points to a string that gets printed
- d the argument is printed as a signed decimal number
- u the argument is printed as an unsigned decimal number
- o the argument is printed as an octal number
- x the argument is printed as a hex number
- f the argument is a float or double and printed as [-]nnn.nnnn, the exact format is determined by a precision string
- e the argument is a float or double and is printed as [-]m.nnnnnn[+/-]Exx, the default precision is six digits
- g the argument is a float or double and is printed as %e or %f, which ever is smaller

Any character following the % which is not a conversion character is printed as is. Be aware that printf() doesn't care if the number of arguments called for in the format string matches the number actually put on the stack. If the string calls for more arguments than were put on the stack then the residue is used, with unpredictable results. Unfortunately, the first release of the Intersoft compiler used a "#" instead of "%". Why they did that totally alludes me.

There is also an input function called scanf(). It operates somewhat similarly to printf() except in the other direction. That is, it allows formatted input from the terminal. I have never used it personally. In fact, I asked a friend who programs in C professionally how often he uses scanf(). The answer was "rarely". However, you may like scanf() so here is a hint at what it does.

```
scanf(frmt, arg1, arg2...)
   char *frmt  A pointer to the format string.
   arg1, arg2  Optional arguments.  As many as are
               called for in frmt.
   returns     The number of arguments successfully
               assigned.

   action      Converts and assigns the input
               stream to the arguments, as
               determined by the format string.
```

The big difference between scanf() and printf() is that all scanf()'s arguments must be pointers to variables that will be filled with the conversions called out for in the format string. Needless to say, if you pass scanf() less arguments than called for in the format string, you can probably kiss the program goodbye. It will use the residue on the stack, which may be pointing into your executable code! How rigorously scanf() requires the input string to match the format string could be dependant on your compiler so I won't say anything more about it here.

CHARACTER FUNCTIONS

The character functions are designed to make manipulation of alphanumeric characters easier.

```
isalpha(c)
   char c;
   returns    TRUE if c is an alpha, FALSE if it
              isn't.

isdigit(c)
   char c;
   returns    TRUE is c is a digit, FALSE if it
              isn't.

islower(c)
   char c;
   returns    TRUE is c is a lower case alpha,
              FALSE if it isn't.

isupper(c)
   char c;
   returns    TRUE if c is an upper case alpha,
              FALSE if it isn't.

isspace(c)
   char c;
   returns    TRUE if c is a "space", FALSE if it
              isn't.  Space is usually defined as
              either <space>, <cr>, <lf>, or <tab >
               but it will probably depend on your
               particular version.

 toupper(c)
    char c;
    returns    The uppercase of c if c is an alpha.
               Otherwise it returns c.

 tolower(c)
    char c;
    returns    The lowercase of c if c is an alpha.
               Otherwise it returns c.
```

STRING FUNCTIONS

There are five string functions that should be considered as basic to any library.

```
strclr(s,n)
   char *s;   A pointer to tha string buffer.
   int n;     The size of the buffer in bytes.
   returns    garbage.  This function is not
              defined in K&R, but should be part
              of your library.

   action     Clears the buffer pointed to by s
              to all NULL's.

strlen(s)
   char *s    A pointer to a string.
   returns    The length of the string pointed to
              by s.  The NULL is not included.

strcpy(s1,s2)
   char *s1, *s2;
   returns    garbage

   action     Copies s2 into s1.  The buffer for
              s1 had better be big enough for s2.

strcat(s1,s2)
   char *s1, *s2;
   returns    garbage

   action     S2 is added onto the tail of s1.
              The buffer containing s1 had better
              be big enough for both both strings.

strcmp(s1,s2)
   char *s1, *s2;
   returns      0  if s1 = s2
              < 0  if s1 < s2
              > 0  if s1 > s2

   action     s1 is compared to s2 until there is
              a difference between the two, or
              either string runs out.
              The difference between the last two
              characters (usually s1 - s2) is
              then returned.  Note that case is
              not ignored, so "foo" and "Foo" are
              not considered equal.  Also be aware
              that the string "aaaaa" is smaller
              than "bb".
```

FILE I/O

There are four functions for handling sequential file. These functions tend to be dependant on the host operating system. Much more so than most others.

```
fopen(filename,mode,fcb)
   char *name;  A pointer to the name string.
   char *mode;  A pointer to the mode string.
   char *fcb;   A pointer to a file control block.
                Used by Dugger.
   returns      A file descriptor (or pointer to
                the fcb) if the open was sucessful,
                otherwise a NULL.
```

Fopen() attempts to open the file whose name is contained in the string pointed to by name. What must be included in the name string depends on whose fopen() you happen to be using. Some versions will not convert the name to uppercase. Others may not assume any defaults.

The mode string contains characters that tell fopen() whether the file is to be opened for reading or writing, and whether it is to be treated as an ascii or binary file (in FLEX).

The file control block pointer, fcb, is only required by Dugger. The fopen() described in "The C Programming Language" doesn't use one. Fopen() supplied by the other 6809 compiler vendors dynamically allocates the necessary bytes. I personally think that this could be a bad practice for FLEX systems. On a memory managed system like UNIX or OS-9 this dynamic allocation poses no problem. With FLEX, it must be done very carefully. The code for one particular fopen() allocated the file control block with the call to the function but didn't deallocate it if there was an error. The block was not deallocated when the file was closed either. You could conceivably write a program that slowly chewed up a lot of memory!

This whole issue of a file control block or file descriptor is EXTREMELY dependant on both the operating system and the particular implementation of fopen(). For FLEX, you can be reasonably sure that the file descriptor is nothing more than a pointer to the actual FCB for that particular file. For OS-9, it would be a "path descriptor". All a file descriptor really really boils down to being is another logical name for an open file which gets passed to the other file handling functions. The exact "type" of the file descriptor depends on your library. I will assume that it is a pointer to a file control block, which in FLEX is an array of chars.

```
getc(file)
   char *file;  A file descriptor for a file that
                was previously opened for reading.
   returns      The next character in the file, or
                the end of file character, EOF, if
                the file is empty or there was an
                error.

   action       getc() attempts to get the next
                character from "file". The detail
                of how this is done is hidden from
                the C program.

                NOTE  If you rewrite getc(), then
                      it is very important that
                      you don't sign extend any
                      characters on return, since
                      0xFF (from a binary file)
                      would get extended into
                      0xFFFF which is -1, the
                      standard EOF character.


putc(c,file)
   char c;      The character to write to the file.
   char *file; The file descriptor of a file that
                was previously opened for writing.
   returns      c if the write was successful, else
                EOF on any error.
```

Note that putc() from some of the libraries may fail to return c, all should return EOF on error. However, putc() is defined as returning c in K&R.

```
close(file)
   char *file;  The file descriptor.
   returns      The file descriptor if the close
                successful, EOF otherwise.
```

So there you have it. These functions will let you do quite a bit in C. You can also use them as a base for writing other functions. For example, say you want to write a function called streq(s1,s2) which returns TRUE if s1 is equal to s2 and FALSE otherwise. You could just pass the parameters on to strcmp() from within streq() and then return the logically inversion of what strcmp() returns.

PEN-IN-MOUTH

Now we come to a part of the column where you get to laugh and I get to cringe. Hopefully this will not be a monthly feature! Seems that I blew a few things in the second column. I noticed most of them. But too late to get the changes into print. 68 Micro Journal reader Jim Howell of San Jose caught one of them. Thank's Jim. Feedback is good, even when it's not complimentary! So here are the bloopers.

mistake #1
An array of fifty int's is declared as
```
   int array1[50];
```
The indices then range from 0 to 49.

mistake #2
The two dimensional array in the function translate() is an array of POINTERS to characters, not characters. I stated that much in words, but the declaration was missing the pointer designator. It should read as
```
   static char *number[2][5];
```

mistake #3
The fixed indices used in the scanning code for the same function were off by one. The code should have read

```
   for (i = 0; i < 5; i++)
      if (match(stg, number[0][i]))
         return(number[1][i]);
```

Once again, if you catch me in an error beyond simple spelling or grammar, then let me have it with both barrels! You will keep me honest and on my toes. You will also be doing a service to the other readers. Please don't take this to mean that you can't send in letter's of encouragement and suggestion also!

WRAP UP

I have recently gotten OS-9 up and running on my system, and Microware's C compiler should be coming along shortly. So far I like what I see of the system. I suspect that OS-9 and C will make a very potent programming environment.

I had a couple of interesting phone calls this week (late March). One was with Introl and the other was with Dugger.

Introl has fixed up a bug and made a few improvements. The bug was the lack of a maximum error count which let the compiler eat up stack space as errors accumulated until it finally crashed. They have made improvements to the error checking of the compiler, improved the error handling of the linker, and added two commands to the librarian that make life a lot easier. Introl is now working on an OS-9 version of their C compiler.

Dugger filled me in on some of the improvements of release #2 over release #1. Types float, double and long have been added. The compiler now outputs position independant code and converts escape characters into byte codes. The FOR and DO WHILE loop statements have been added, as have the logical NEGATE, OR and AND operators; "!", "||" and "&&" respectively. The complete set of assignment operators are now available too. The compiler did get bigger. You will need 32K to compile any reasonably sized program. Dugger is also working on an OS-9 version of their compiler.

Next month we'll go over ways to make life easier when you you're using a compiler without a relocating assembler and linker, and what some vendors have done to help you along these lines. I will also include a crude bench mark of the Dugger, Intersoft, Word's Worth and Introl compilers based on the Eratosthenes Sieve prime number algorithm. This code for this algorithm was taken from the September, 1981 issue of BYTE magazine and then modified to run on the Dugger subset of C.

COLOR User Notes

Robert L. Nay
5900 Cassandra Smith Rd.
Hixson, Tn. 37343

QUICK LOOKS:

We have received a couple of items from Pete Stark of **STAR-KITS** (see Advs. in this Magazine) for the Color Computer. Pete has set his excellent **SPELL 'N FIX** program up to run on the EXATRON FLEX Conversions (i.e., moved it above the Display Screen Memory area). He also now has it available in Radio Shack Disk Format, for those that don't have the FLEX Operating System. And, of course, the normal FLEX9 Version of the program will run without modification on the 64K/Radio Shack Disk Controller FLEX Conversions being sold by **DATA-COMP** and **Frank Hogg**. This is a "request the information" type of program that is extremely easy to use. The programs' "prompts" are clear and easy to understand, and the documentation for the program is excellent. See page 31 of the Feb '82 issue of **'68' Micro Journal** for a full review.

The other product we have been using from **STAR-KITS** is his **HUMBUG** Super Monitor for the Color Computer. This is available either on Tape or Disk (R. S. Format - $39.95) or in a ROM Pak for $69.95. Again, the Manual is

excellent, and includes a full Source Code Listing of the program. **HUMBUG** contains 37 different two-letter Commands which allow the normal Memory examine and change operations including ASCII and HEX Dumps, Changes, Examines, Checks, Tests, and even MV for Memory View. You can also Analyze, Load, Punch (write), Save, etc. with Tape; play with the Registers; operate as a Terminal in Full or Half Duplex; handle the Printer and Baud Rates; etc., etc., etc. It provides a Disassembler Dump, Fill and Move Memory, and Single Stepping through programs with full Breakpoint features. A couple of unique Commands I haven't seen tell you Where Humbug is (WH) and allows you to Move Humbug (MH). These are extremely handy when running a Memory Test; you can test up TO Humbug, move Humbug down, and test on up: NICE. In general, it is an **EXCELLENT** Monitor; we will have a full review on it in a later issue.

Another OUTSTANDING product we have been using is a **Lowercase Adapter** for the Color Computer from **Micro Technical Products, Inc.** Again, see the Adv. in this issue. This is an Adapter Board that contains a ROM and the circuitry to automatically switch it on when you select Lower Case Characters (with <SHIFT-0>); no wiring or circuit mods are required. The MC6847 Video Display Generator is removed from it's socket and the Adapter Board plugs into the Socket just emptied, then the 6847 is reinstalled in a socket provided on the board. The pins that go into the normal VDG Socket are formed like normal IC Pins and long enough to seat solidly; there will be no problems with the socket pin receptacles being bent out of shape from forcing thick pins into the sockets, or having the Adapter Board "pop out" because you just couldn't press it down deep enough. The Character formation of the letters is also excellent. Another SUPER Feature of this Lowercase Adapter is that there are two switches on the board; one switches the ROM on or off (when off, you are back to the normal Display Screen), and the other switch INVERTS the colors on the Display. We immediately set ours for a Black Screen with Green Characters and LEFT IT THERE - a true "Green Screen" Display. This is another **EXCELLENT** Product for the Color Computer, and is competively priced at $75.00. Order them from **Micro Technical Products** or **DATA-COMP**; I'm sure you won't be disappointed.

I have received several requests for information on some type of board that can be plugged into the ROM Pak Slot to allow wiring your own circuit for the Color Computer. We received a unit called the **PROTOPACK** from **Atomtronics** (3195 Arizona Ave., Los Alamos, NM. 87544) the other day. This is a PC Board about 4" sq. filled with holes on a 0.1" grid. One side of the Board is completely plated with the plating relieved around each hole which acts as a Ground Plane; the other side has plating for each hole. The Board "Edge Connector" is Gold Plated, and the +5 VDC Line (from pin 9) is continued around the Board to provide a Powor Buss. The documentation provides a listing of the ROM Connector's pinouts and some suggested circuits for wiring a 6821 PIA, EPROMs, 2114 Static Memorys, etc. Also included is a discussion of the Color Computers' Power Supply limitations. Finally, a Plastic Case is available which houses the Board to protect it from the "elements" (paperclips, staples, resistor leads, etc.; all those things that are attracted to a Circuit like magnets). The Board sells for $24.95, or $32.95 with the plastic case. A "Tinkerers Dream"!

COMPUTERWARE has been providing excellent 6800 and 6809 support for a while, and they have numerous excellent products available for the Color Computer. Their product line includes both Hardware and Software (I'm sure you have seen their Advs. in this magazine). They also provide an informative "newsletter" about their products; if you are not on their mailing list, drop them a line. I would like to mention a couple of their products here. The first is a product which should be a part of every Color Computer Owners Inventory; the **COLOR DATA ORGANIZER.** This is a small "Data Base Management" type of BASIC Program on Tape which allows 2 Alphanumeric fields and 2 Numeric fields of information access and storage. Its' capabilities are similar to Radio Shacks "COLORFILE", but, since it is written in BASIC, you can "mold" it to your own needs. It can also be converted to Disk when you get that capability. Besides being able to change Printing Formats, Display Formats, Record Formats, etc., it also provides a good example of the use of BASIC in both Programming and for something besides playing games (which is not that BAD).

The other Product from **COMPUTERWARE** is the first "Disk Based" Textprocessor, in Radio Shack format, to hit the office; **COLOR SCRIBE.** COLOR SCRIBE comes on Disk and requires 32K of memory. The package sells for a very reasonable $49.95 and includes the Disk and an exceptional, attractive Manual. I call it a "Textprocessor" because that is what the Program is; an excellent "Line Editor" with a full range of "Formatting Commands" that are inserted at the beginning of a line of text which control the output of the Text when it is "Processed". This formatted output can be directed either to a Printer or to the Screen. **COLOR SCRIBE** has two modes while inputting text; the INPUT Mode and the COMMAND Mode. Text is entered in the INPUT Mode, which furnishes automatic line numbers: these are used for reference while manipulating text only; programs such as BASIC which need line numbers must have them entered following the Editors Line Number. The full range of Text Manipulation is supported including extras like "selective change", Macro Commands (which allow defining a command or series of commands which are activated by a single key-press), etc. The Formatting capabilities support such advanced features as Headers and Footers with automatic Page Number insertion, Fill and No-Fill Modes, Centering, positive and negative indentations, etc. The full range of Tape and Disk Handling commands are supported, as are a few "Special Color Computer" commands such as "RV" for Reverse Video to make the Display more 'useable' when it contains mostly Capital or Lowercase letters. This program will probably be the "Most Used" program you own because it is used for Programming, Word and Text Processing, etc. An excellent program.

MORE on R. S. DISK CONTROLLER with FLEX

By now both **DATA-COMP** and **FRANK HOGG LABS** should be shipping their FLEX9 Conversions for operation with the Radio Shack Disk Controller. This package should prove to be the "Standard" Operating System for the Color Computer for several reasons.

1. This system allows maximum use of the 5 1/4" Disk Drives available to the normal Color Computer User. The WD1793 Disk Controller Chip used in the Radio Shack Disk Controller allows Double Density storage on the Disk. Both Conversions offer Double Sided capability through the use of the Drive Select 3 Line. The combination of these two factors allow 1404 Sectors of storage on a 40 Track, Double Sided Disk Drive, or just under 400,000 (Formatted, meaning USEABLE) bytes of storage. With the "state of the art" on Disk Drives at the present time, this is probably the most "dollar efficient" and reliable mass storage available for the average Color Computer User.

2. This system allows COMPLETE compatibility with the Radio Shack Disk System (in fact, you really have TWO COMPLETE Disk Operating Systems). As good Software begins to show up for the Color Computer, you can have "the best of both worlds". The FLEX System allows the use of the large amount of good Business Software available for serious use on the Color Computer, and the Radio Shack System allows the use of that type of Software as it begins to arrive.

3. I haven't seen the **FRANK HOGG LABS** System yet, but the **DATA-COMP** Package eliminates several of the

Color Computers' natural limitations. The full 128 ASCII Character Set is provided through the use of a "CONTROL" Key and "User Definable" Keys (an "ESCAPE" key is also provided), and four different Display Screens are incorporated, including 42x24 and 51x24 Display Formats with normal lowercase. Both of these Screens are very useable, and considerably reduce the Display limitations of this computer.

4. The General Software available for the FLEX Operating System includes an extremely POWERFUL and FAST Business BASIC in TSC's XBASIC, an excellent **Macro Assembler, a powerful Line Editor** (Computerwares' COLOR SCRIBE, mentioned previously, is a small version of this system), and an EXTREMELY POWERFUL **Text Processor**, to mention a few of the Software Products that form the basis of a Powerful Computer System. For the Programming types, I have been running TSC's **PASCAL**; SWTPC's **PILOT**; Ouggers', Introl's, and Words Worth's C **Language**; all "straight out of the box", so to speak. In general, if the Program uses the normal FLEX I/O Vectors, it will run on this system, if you have enough Disk Storage capacity.

All is not roses, however. First, we have been looking into the RAM situation these past several weeks. Out of **FIVE (5)** sets of the Radio Shack 32K RAMs that we have checked, only one set came up as good 64K Chips when the unit was first modified, and they didn't last a week. **EVERY set** has had failures within a week and all but one set still averages losing a Chip about every week. In short, **WE have not seen a set of Radio Shack 32K RAMs that were GOOD 64K Chips.** We have been using the TI4164 series Chips with good success on the Color Computer. The Refresh requirements of this RAM Chip are different from the normal 4116 or Motorola 6664/5 type chips, but they work fine since there are plenty of extra Refresh Cycles provided with the Timing used in this Computer. **MEMORY** is NOT the place to "save a buck" where reliability is concerned; **MAKE SURE THOSE RAMs ARE GOOD 64K Chips.**

The other problem you may face is in the Disk Drives. Again, "you pays your money, and you takes your chances". The first problem that may appear is in the area of the "Double Density". **THE DISK DRIVE DETERMINES whether your system will work with Double Density** when using these FLEX Conversions with the Radio Shack Disk Controller. The **SOFTWARE** supports Double Density, but if the Disk Drives' Read/Write Head and Electronics are marginal, you are heading for trouble with Double Density. Most of the Late Model Disk Drives can handle Double Density with no problem because of recent advances in technology, primarily in the Head Design itself. The Radio Shack Disk Drive for the Color Computer is an excellent product and will give no problems in this area; TEAC, Tandon, Qume, etc., all have units that operate reliably in the Double Density mode. Again, you get what you pay for.

If you want to make use of the Double Sided capabilities of these FLEX Conversions, you will need to replace the normal Radio Shack Disk Drive Cable because they accomplish their "Drive Select" with the Drive CABLE, not in the Disk Drive like every one else. If you look at the Drive Connectors on a Radio Shack Cable, you will notice that several connector pins are missing. They "Select" a Drive by only installing the appropriate Drive Select Line to that particular Drive; e.g., the connector for Drive 0 **ONLY has a pin installed for the DS 0 Line**, the connector for Drive 1 **ONLY has a pin installed for the DS 1 Line**, etc. Their **DISK DRIVES** have **ALL OF THE DRIVE SELECT LINES ENABLED.** Their Drive 0 is also SPECIAL; it has the **Terminating Resistor pack BUILT IN.** That is why that Drive ALWAYS goes in the LAST DRIVE POSITION. So; we have a Cable that does **NOT** have a **DS 3 Line**, on Drives that **DO NOT** have "Drive Select" jumpers.

To use the **Double Sided Drives** with these Conversions, you must provide the OS 3 Line to ALL of the Drives, because IT is used as the Side Select Line. **DATA-COMP**, for one, has Disk Drive Cables available with ALL of the lines installed (be sure to specify this type of Cable when ordering a system from ANYONE). Also, if you are using one of these cables on a **Radio Shack DISK DRIVE**, you will have to MODIFY THAT Drive by cutting the PC Lands for all of the Drive Select Lines EXCEPT the one being used (all other Disk Drives provide a "Jumper Block" to accomplish this).

Now that we have the OS 3 Line going INTO the Drive, we will have to MODIFY the Circuit Board in the Disk DRIVE itself. Since no Disk Drive Manufacturer had the foresight to provide a "Jumper Block" to enable using the OS 3 Line for Drive Select (it WOULD take a HIGH QUALITY Crystal Ball to foresee this requirement; we only run into these kinds of problems when working with Radio Shack), we must provide that capability ourselves by replacing the Side Select input with the OS 3 Select input. Normally, this is accomplished by cutting the Side Select PC Land and jumping the OS 3 Line to the Drive side of this cut. All that is left is to determine how to set up the rest of the jumpers; for instance, on the TEAC Model 50C's, we had to enable the "MX", the appropriate "DS" line, and the "HS" line, to get them to work Double Sided. Next month, we'll try to summarize the mods to some of the normally used Drives.

The FLEX Conversions by **DATA-COMP** and **FRANK HOGG LABS** provide the Software to make the Color Computer a powerful Computing System, but YOU will have to provide the RAM and Disk Systems to allow maximum utilization of this potential.

--- RLN ---

NIXON VID Brd

NIXON NES002A CRT Display

For those users who operate with the Motorola Exorcisor™ systems this review should be good news. For as long as I can remember, if you were on a Motorola system bus you did not have as wide a variety of vendors (boards, software, etc) to choose from. The Motorola components are fine but the price and sometime availability were something else. No doubt the Exorcisor™ system is a fine developement system, but then there are those of us who have opted to use ours for applications as well. The choice of software and support hardware becomes somewhat more narrow. Also the price is a major factor. The NIXON Engineering Company offers a full 'Exorcisor™' developement system, and at very attractive prices.

NIXON Engineering Systems are packaged in two very attractive cabinets. The disk drives are housed in one rugged and stylish 'Moduline" industrial grade cabinet, the computer in a matching cabinet.

The NIXON Exorcisor™ system consists of 5 or 8 inch disk drives, a 6809 CPU card, 56K of RAM and I/O. The standard system includes a 10 card cage. Expansion room is more than sufficient for additional ports, not including the normally supplied two serial ports available at the back and the two parallel ports that are also available. The power supply is a 'Boschert' switching power supply of rugged design. The system is truly an Exorcisor™ compatable system both from the hardware and software angle. The price is certainly right, a complete system with dual double sided 5 inch drives sells for less than $5,000.

Before detailing the video board a few things merit mention. The disk controller, which is a double sided configuration, can drive four SA-800 and three SA-400

type disk drives with selection of all seven always on line, sells in the $395 range as does the video board. Compare the features and then the price and the NIXON system becomes very attractive (a savings of one half to two thirds).

The NES002A CRT DISPLAY MODULE appears to the system as a 4K static memory device. It generates 84 characters X 24 lines with an optional 80 X 25 display. all 128 characters are formed in a 7X9 dot matrix. Also 16 different combinations of background/underline character positions allow a high degree of system display flexibility. It drives both composite and separate syncronizations devices. The display by being synchronized to the system clock provides a glitch-free display between the CPU and system display.

FEATURES

1. Upper and lower case.
2. Shifted lower case (extenders) for (J,y,g,p etc.)
3. Background/underline control for each character using upper 2K memory page for control
4. Upper 4 bits of control page available for other software use
5. TTL level horz. and vertical sync outputs (positive true)
6. 2.5v P-P video out with adjustable halftone level
7. Clock normal 1.0 MHZ 2.0 MHZ optional

Addressing: Base address is switch selectable to any one of the 16 4K positions in the 64K field.

The video board is now available and adaptable to the Motorola VERSA™ bus by using the VERSA bus Exorcisor™ adaptor.

The very popular FLEX™ disk operating systems is also available for this system. NIXON ENGINEERING furnished a set of overlay drivers to enable the system as a full FLEX™ controlled computer, great for running all those applications and other software advertised in 68 Micro Journal.

Additional information and availability can be secured from:

NIXON Engineering Co.
578 Menker Ave
San Jose, CA 95128
a/c (408) 371-4573

2 MHz PROM

SWTPC EPROM PROGRAMMER

It came to my attention some few months back that SWTPC had developed additional software for their EPROM programmer. This was an in-house project and also is furnished with later versions of their FLEX™.

Realizing that there are many of you out there who have this unit but do not have the latest software to use it at 2 mhz, or utilize most of its other nice features, I called Dan Meyer, President of Southwest Technical Products and after discussing it with him he agreed to allow me to supply you with the software in binary on the 68 Micro Journal 'disk sales' offer found elsewhere in this issue. We all say 'THANK YOU' to Dan Meyer and the folks at SWTPC for their willingness to allow us to furnish you their software on a near cost basis.

The programs allow you to read, write to disk 2716, 2532 EPROMS and also checksum each one. And even more important they allow programming at up to 2 mhz both 2716 and 2532 5v type EPROMS.

We have prepared a short instruction sheet on their use and will supply one with each disk requested.

Please bear in mind that because of the individual handling of each of these type of request we will supply only as we have it and the instructions assume you are confident in your computer and it's software capabilities and it's operation.

Thanks again Dan and all you fine folks down at SWTPC.

DMW - - -

RECEIVE BREAK CIRCUIT

J. Gary Mills
1019 Weatherdon Ave.
Winnipeg, Manitoba
Canada R3M 2B5

This article describes a simple addition to a serial interface card which is used to detect the signal generated by the break key on a keyboard.

The break signal from an ASCII keyboard is simply a long space signal. During transmission of normal characters, short spaces occur, but there is always a mark level at the end of each character. The break signal is generally timed (230 ms is a common choice), but some terminals transmit a continuous space as long as the key is depressed. The break signal can be detected by the computer hardware, and used to generate an interrupt signal.

With the 6800 series of MPUs, an interrupt is initiated by applying a logic low to the NMI (Non-Maskable Interrupt) line. The MPU then completes execution of the current instruction, saves the register contents on the stack, and begins execution of the NMI service routine.

The circuit shown in Figure 1 is based on a CMOS 12-stage binary counter. The counter is reset during a mark and counts up from zero during spaces. When a long space occurs, the count reaches 256 and the Q8 output goes high. A single inverter is required to drive the NMI line. This should be an open-collector inverter if any other devices are connected to the NMI line. The nor gate causes the counter to remain at a count of 256 as long as the break signal is present. When the mark level returns, the counter resets to zero.

I am using MIKBUG version 2.0 in my system. This monitor responds to an NMI interrupt by printing a 'B' on the terminal and then displaying the register contents from the stack. I can then use all the commands of the monitor to continue debugging the interrupted program. Some earlier versions of MIKBUG do not have this feature. However, most versions maintain the address of the NMI service routine (two bytes) at A006 in the system RAM. This means that you can include a service routine in your own program and place its address at A006. The actions performed by the service routine are up to you, but generally it is a good idea to begin by reading the ACIA to clear out the garbage characters generated by the break. There are only a couple of ways to return to the interrupted program. One way is to set a flag to indicate the break and then issue an RTI instruction to return to the point of interruption. The other way is to reset the stack pointer and then branch to a different portion of the program.

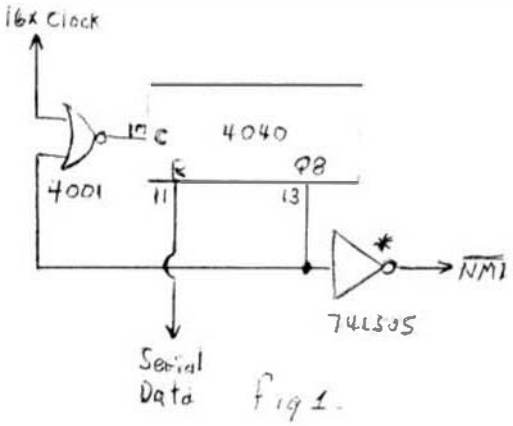


Fig 1.

PRODUCT REVIEW

DATA SYSTEMS-68 DRAM-64K MEMORY BOARD

:by the '68' Micro Journal Staff

We have been evaluating a product from the heart of the "Eastern Silicon Valley" (Central Florida) the last couple of weeks that will warm the hearts of many of the SS-50 type 'experimenters'. In this case, the 'ProducER' goes by the name of **DATA SYSTEMS-68**, located at 2316 Diversified Way, Orlando, FL. 32804 (see Adv. this issue). The 'ProducEE' is a 64K Dynamic Memory Board, appropriately named the **DRAM-64K**, available either with or without "population" (parts to you newcomers). The Board is extremely well made, consisting of a 1/16" G-10 Glass Board measuring 5 1/2" by 9" for the 50 pin slot. It is a Double Sided PC board with Plated-through Holes, Solder Masked on both sides, and a fully Silkscreened Component Layout on the top. The PC work is professionally done, and appears to be 2 oz. Clad with a 'healthy' pre-tinning. The component layout is 'spacious' due to the use of LSI's for Refresh, and the Silkscreening provides the parts VALUE, not reference (e.g., 22K, 74LS245, etc.), which means the complete Board could be assembled without referring to the instructions (which are very complete).

The Board is designed around the inexpensive, but "old reliable", 4116 Memory Chips; and the Motorola MC3242 - MC3480 LSI pair to keep them Refreshed. It also eliminates any timing problems by using the 0447-0250-01 Delay Line Chips instead of adjustable one-shots. This design yields several advantages: It allows a relatively spacious layout when compared to boards using TTL for Refresh; it uses parts with proven reliability and low cost; and it eliminates any adjustable components, increasing reliability that much more. Just install the parts (correctly) and it is ready to go.

The **DRAM-64K** Memory Board is a high-quality PC Board which can be populated primarily from parts that inhabit most any "parts bin". The 'specialized' parts such as the two Motorola LSIs, the Delay Lines, and the Molex Connectors are available directly from **DATA SYSTEMS-68**, along with the Board. The Memory Board is designed to run on both 6800 and 6809 Systems at 1 MHz, and includes an eight position DIP switch for enabling memory in 2K increments along with extended addressing capabilities. Now for the best part; the **DRAM-64K** Bare Board costs a whole **$39.95**. Allowing around a hundred dollars for the parts, you could put together a bank of 64K Memory for around $150 (remember what that much memory cost a couple of years ago?). **DATA SYSTEMS-68** has several more projects "in the works"; if this is any indication, they are going to be "warmly welcomed" by the SS-50 crowd. An outstanding Product and an outstanding Price.

-- RLN --

DATA SYSTEMS '68'

Dedicated to the Abandoned Hobbyist

In 1979, Jack Morrison, president and owner of Central Florida Circuits (CFC), a company specializing in the manufacture of high-quality prototype and limited production printed circuit boards, and his personnel director, Tony Montgomery, began to explore the avenues available to them in computer sciences.

Mr. Montgomery, a 6800 user, brought his own Southwest 6800 in and demonstrated the usefulness of a computer to Mr. Morrison on both the business and personal level. Mr. Morrison got hooked on the old "Space Voyager" game. They decided to set aside ample space and funds to further study the 6800: what had already been done, what was being done and, more importantly, what could be done in the future. Contact was made with the local 68XX users' club. A few of the more sophisticated engineers and hobbyists in this club began feeding ideas and suggestions to CFC. The main objective was to make the 6800/6809 a more versatile system.

In May of 1980, the project became so time and labor intensive that CFC formed its own research and development department to handle the load. During the next 14 months, R & D built an extremely large inventory of computers, PC boards, programs and test equipment. Fortunately, any custom PC boards that were required for this research were manufactured in house by CFC.

In July 1981, R & D had grown to the point where it required its own building and staff. Several of the engineers and hobbyists who had been working with CFC all along were hired and the core of what would soon be Data Systems '68' was formed.

Data Systems '68' offers PC boards for the 68XX user of unequalled quality and design. All DS-68 boards are solder masked on both sides. All nomenclature is silk screened for ease of assembly. Data Systems '68' documentation is the best in the industry (over ten full pages of documentation on some boards!). With 17 boards available this year and new products still in the testing stages, Data Systems '68' is the brightest new name in the world of the 68XX user. A new main frame system will be unveiled at the Atlanta ComputerFest on June 12. The new Data Systems '68' stand-alone terminal will also be shown. It features a 12" CRT, 1.5 megabites of storage and 56K of usable memory.

Data Systems '68' is a company dedicated to you, the 68XX user. They welcome ideas, suggestions and opinions from all. In the months to come, watch for more exciting products from this dynamic new company.

tFORTH

Talbot Microsystems
1927 Curtis Ave
Redondo Beach, CA 90278
(213) 376-9941
April 9, 1982

Don Williams, Publisher
'68' Micro Journal
Hixson, TN 37343

Dear Don,

Many readers probably have wondered whether the FORTH community is just blowing hot air over the advantages of FORTH, or whether there are any facts to substantiate the claims that FORTH produces code which is both compact and fast. On the issue of speed, there are two points to be addressed: speed of the programmer to develop code, and speed of execution of that code.

On programmer speed, there is to my knowledge no hard data, just a common "knowledge" in the FORTH community that productivity with FORTH is high. From personal experience, I know that for laboratory work involving one-time throw-away code I routinely do in 10 seconds what would require me 5 to 15 minutes to do via the usual editor - assembler/compiler - load - execute route with assembly language or BASIC, PASCAL, or FORTRAN. I have had customers of firmFORTH comment that they were able to develop 8-16k target rom codes 3-10 times faster than previously using assembly. PASCAL and other language proponents will immediately point out similar gains when moving to their high level language, but these users generally comment that those other languages do not provide the flexibility, power, or interactivity of FORTH, and that they need compact and fast code which those other languages do not provide; i.e, the requirements of the job force the decision to be assembly or FORTH.

On the actual code execution times and memory requirements, there is firm information. The prime number benchmark of Anderson (July '81 MJ) has been used to compare the various 6809 implementations of PASCAL (Nov '81, and April '82). I have coded the equivalent in FORTH and present the results in the accompanying table and graphical representation. (Note: the April '82 times for OS9 have to be doubled for proper comparison: it is improper to compare software using hardware with different cpu speeds! My table has all times adjusted to a common 1Mhz cpu). There are some subtle aspects which I do not have sufficient information to discuss; e.g., the firmFORTH code is completely self contained and rommable, but I do not know about the PASCAL's. I suspect that some of them are also, but it is also likely that some of the PASCAL's use host operating system calls to do terminal I/O. If I use FLEX IO calls for all terminal I/O instead of internal runtime code, the firmFORTH total byte count drops to 1075 bytes!

Conclusions: FORTH is significantly faster than any current p-code 6809 implementation of PASCAL, and only about 25% slower than native code PASCAL's. The code produced by firmFORTH is much much smaller than most PASCAL's. In fairness, it must be pointed out that firmFORTH is integer only, and so without extensions is is generally regarded as being useful mainly for applications such as instrument controllers, data acquisition, and process control. Of course, those are generally the situations where compact rommable code is of importance anyway.

As a point of interest, for the same benchmark run with the full tFORTH compiler/interpreter rather than the runtime package, the number of user bytes and total bytes are 382 and 9887 (speed is same as firmFORTH), Tiny PASCAL has been written in FORTH (PASCAL compiles into the internal FORTH code) and for this the parameters are 716 user bytes, 16,826 total bytes, and 192 seconds. 17k looks high compared with the numbers in the table until you remember that this 17k includes the full FORTH compiler/interpreter as well as the PASCAL compiler/interpreter and then the application itself! I do not know the sizes of the various PASCAL compilers, but I dare guess that they are larger than this! The PASCAL-in-FORTH compiler was never intended to be a production tool (straight FORTH is better), so it has had no optimization efforts applied to try to speed it up. It does, however, provide a very unique and powerful learning tool because you can easily intermix PASCAL, FORTH, and assembly code, and test code immediately.

QUICKSORT

by HL Harkness

Here is the sort routine I put off for later back when I did my linking loader. Since, at this writing, I have recieved no orders for my loader, anyone who sends for it will get the version with the sorted symbol table feature.

I agonized for a while on just what algorithm I was going to implement for my sort. For short lists, namely any that are likely to fit into main memory of a micro system, nearly any old sorting algorithm is OK. I started to do a bubble sort, but when I got started, I saw that it was actually a little more complicated than some of the better sorts.

I read through the section on sorting in Software Tools, by Kernighan and Plauger, and decided to do the quicksort by C.A.R. Hoare instead.

Quicksort is really kind of neat, but I wasn't sure I understood the Algorithm. Knuth's Sorting and Searching, (Vol. 3 of his series, The Art of Computer Programming) was a little more help. Both of these references gave very detailed non-recursive routines for quicksort. Since the 6809 is well-endowed for recursion, I decided to use recursion anyway. I studied both references at great length until I convinced myself that I could write it.

Quicksort had yet another attraction for me. I first encountered it back in school in a class on LISP taught by none other than Dr. Siklossy'. I failed that course. Miserably. And Quicksort was one of the things that gave me the most trouble. I just couldn't convince myself that such an algorithm would even work -- I certainly couldn't make it work in LISP. So, to settle an old score, I wanted to prove I could do it.

It seems so simple. All you do is pick an arbitrary entry in the list, and

re-arrange the list so that all entries which are larger come after that entry, and all entries which are smaller come before it. Then you repeat that process for both of the lists on either side of that entry. When you get to a sub-list which has only one entry, that entry is already in the proper location in the list. (Trust me)

If you are like me, however, the recursion part is difficult to follow. I think perhaps that is because the very first language I learned was FORTRAN. The recursive description in K&P is only ten lines long, and yet it must have taken me three days to get it straight.

```
SUBROUTINE QUICK(V,I,J)
IF(I>=J)
  RETURN
PARITION THE ELEMENTS V(I)...V(J) SO THAT
  V(I),V(I+1)..V(K-1) <= V(K) <= V(K+1)..V(J)
  FOR SOME K SUCH THAT I <= K <= J
CALL QUICK(V,I,K-1)
CALL QUICK(V,K+1,J)
RETURN
END
```

I've plugged Software Tools and Sorting and Searching (vol 3) a couple of times now, yet they deserve another mention. Anyone seriously interested in programming should have both of these books, so if you don't, you should drop everything and run down to your local computer store and get them both. And I do mean all three of you.

As it turned out, I had absolutely no problem with the recursion. It was so simple that I even went back and did the extra calculations to make sure that I always sorted the shortest sublist first (see comments in listing). The things I did have some problems with were an instruction that didn't quite do what I expected, and some pointers which were hard to keep straight. In real life, I work in a 16-bit world, and the mixture of 16-bit and 8-bit quantities in '09 code sometimes throws me.

In the integration process, I uncovered a mistake in SCOMPR, and I changed the call to SORT in NTERPS slightly, so I am including the altered listings in addition to the three routines written for SORT.

The loader's hierarchy chart given in my last installment is now amended to include:

```
NTERPS  Interpass process
  SORT  Sort symbol table
    SCOMPR  Compare strings
    EXCHAN  Exchange strings
      COPY  Copy string
```

The next logical improvement to be made on RLOAD is to organize the load map to make it take up less paper and make it easier to read. However, I am now proceeding to write some data base routines I had in mind when I wrote SORT. After all, the only reason for having a tool like RLOAD is do some useful work with it. When I get tired of the way it works, I will sharpen it some more...

Please address orders or inquiries to Word's Worth, Box 28954, Dallas, Texas 75228.

```
NTERPS                          7-19-81 TSC ASSEMBLER
Author: BL Harkness, Placed in Public Domain, 1981.


0000                            MODULE 'NTERPS  - RLOAD Vers 2.1 '
                        * * * * * * * * * * * * * * * * * *
                        * INTERPASS PROCESS
                        *
                        * In the final version, this module will Produce a
                        * sorted symbol table listing and
                        * will point out undefined externals. In the interest
                        * of brevity and time, the intial (bootstrap) version
                        * will merely dump the table.
                        *
                        * . . . . . . . . . . . . . . . . . .
                        *
                        *REV2.1 Changed calling sequence of SORT to use length
                        *       of key instead of column numbers 15JUL81 HLH
                        *
                        * . . . . . . . . . . . . . . . . . .
0000                            ENT     NTERPS
0000                            EXT     SYMTAB
0002                            EXT     SYMTABEND
0004                            EXT     SORT
                        *
0006                            EXT     PSTRNG
0008                            EXT     OUTHEX
000A                            EXT     PCRLF
                        *
                 0006   SLEN    EQU     $6          STRING LENGTH
                 0008   RECLEN  EQU     $8          ENTRY LENGTH
                 003D   EQUALS  EQU     $3D         =SIGN
                        *
                 000C   NTERPS  EQU     *
000C AD   9C FB                 JSR     [PCRLF,PCR]
000F 30   8D 0058               LEAX    MSG,PCR
0013 AD   9C F0                 JSR     [PSTRNG,PCR]
0016 AD   9C F1                 JSR     [PCRLF,PCR]
                        * Sort the symbol table
0019 AE   8C B6                 LDX     SYMTAB, CR
001C 36   10                    PSHU    X
001E AE   9C DF                 LDX     [SYMTABEND,PCR]
0021 36   10                    PSHU    X
0023 86   08                    LDA     #RECLEN
0025 36   02                    PSHU    A
0027 86   00                    LDA     #0
0029 36   02                    PSHU    A
002B 86   06                    LDA     #SLEN       *REV2.1
002D 36   02                    PSHU    A
002F AD   9C D2                 JSR     [SORT,PCR]
                        * Print the symbol table
0032 AE   8C CD                 LDX     SYMTAB,PCR Point to top of table
                 0035   PRSYM   EQU     *
0035 36   10                    PSHU    X
0037 31   8D 0049              LEAY    STRING,PCR
003B C6   06                    LDB     #SLEN
                        * Copy the symbol to output buffer
                 003D   COPY    EQU     *
003D A6   80                    LDA     ,X+
003F A7   A0                    STA     ,Y+
0041 5A                         DECB
0042 26   F9                    BNE     COPY
                        * Terminate the string.
0044 86   3D                    LDA     #EQUALS
0046 A7   A0                    STA     ,Y+
0048 86   04                    LDA     #EOS
004A A7   A4                    STA     ,Y
004C 30   8D 0034               LEAX    STRING,PCR
0050 AD   9C B3                 JSR     [PSTRNG,PCR]
                        * Get the address of the symbol
0053 37   10                    PULU    X
0055 C6   06                    LDB     #SLEN
0057 3A                         ABX
0058 AD   9C A0                 JSR     [OUTHEX,PCR]
005B 30   01                    LEAX    1,X
005D AD   9C A8                 JSR     [OUTHEX,PCR]
0060 30   01                    LEAX    1,X         Now pointing to next entry
0062 AC   9C 9B                 CMPX    [SYMTABEND,PCR] If there is one
0065 2D   CE                    BLT     PRSYM
                        *
0067 AD   9C A0                 JSR     [PCRLF,PCR]
                        *
006A 39                         RTS
                        *
006B 20 2A 2A 2A        MSG     FCC     ' **** SYMBOL TABLE *****',EOS
006F 2A 20 53 59
0073 4D 42 4F 4C
0077 20 54 41 42
007B 4C 45 20 2A
007F 2A 2A 2A 2A
0083 04
0084                    STRING  RMB     $8
                        *
008C                            ENDMOD
                                END

0 ERROR(S) DETECTED
```

```
SORT                                    7-19-81 TSC ASSEMBLER
Author: ML Harkness. Placed in Public Domain, 1981.

 0000                           MODULE 'SORT     - BLOAD Vers 2.0 '
                              * * * * * * * * * * * * * * * * * *
                              * SORT - GENERAL-PURPOSE SORT ROUTINE
                              *
                              * Sorts a table of equal length entries of source
                              * strings by keys specified by column positions.
                              *
                              * Inputs on U stack:
                              *   U+0 Key length
                              *   U+1 Start column
                              *   U+2 Record length
                              *   U+3 PTR to end of table (past last element)
                              *   U+5 PTR to beginning of table
                              * Output: U cleaned up
                              *
 0000                                 XDEF    SORT
                              *
 0000                                 XREF    COPY
 0002                                 XREF    EXCHAN
 0004                                 XREF    SCOMPR
                              *
                      0006    SORT    EQU     *
                              *
                              * Since U is used by the Quicksort routine for passing
                              * parameters to its recursive calls, we need to save
                              * those values which don't change in local storage.
                              * This will also leave the U stack in the proper state
                              * for the first 'call' to QUICK, which is done by just
                              * falling through.
                              *
 0006 37   06                        PULS    A,B
 0008 A7   8D 0008                    STA     KEYLEN,PCR
 000C E7   8D 0005                    STB     STARTC,PCR
 0010 37   02                         PULS    A
 0012 A7   8D 0000                    STA     RECLEN,PCR
                              *
                              * Adjust the end pointer to point to last element
                              * instead of past end of table. (U points to end pointer)
                              *
 0016 EC   C4                         LDD     ,U
 0018 E0   8D 000A                    SUBB    RECLEN,PCR
 001C 82   00                         SBCA    #0
 001E ED   C4                         STD     ,U
                              * * * * * * * * * * * * * * * * * *
                              *
                              * QUICKSORT ROUTINE
                              * Inputs: U+0-upper partition limit
                              *         U+2-lower partition limit
                              * Outputs:U stack cleaned up
                              *
                              * Reserves 6 bytes of temporary storage on U stack
                              *
                              * Refer to KAP, 111-116 and
                              *           Knuth, vol 3, 114-123
                              * * * * * * * * * * * * * * * * * *
                              *
                      0000    LOWER   EQU     0       Temp storage for lower bound
                      0002    UPPER   EQU     2       Temp storage for upper bound
                      0004    PTREND  EQU     4       End of partition
                      0006    PTRBGN  EQU     6       Beginning of partition
                              *
                      0020    QUICK   EQU     *
 0020 33   5C                       LEAU    -8,U    Reserve temporaries
                              *
                              * See if we are done with this partition by
                              * checking for UPPER<LOWER
                              *
 0022 10AE 44                         LDY     PTRBGN,U
 0025 10AC 46                         CMPY    PTRBGN,U
 0028 102F 0085                       LBLE    QEXIT
                              *
                              * Initialize upper and lower pointers
                              *
 002C AE   86                         LDX     PTRBGN,U
 002E 10AF A2                         STY     UPPER,U
 0031 AF   C4                         STX     LOWER,U
                              *
                              * Now for the hard part -- partitioning.
                              *
                              * PY has the pointer to the end of the list to be sorted.
                              * This pointer will be used as the pivot pointer.
                              *
                      0033    PART    EQU     *
                              *
                              * Point PY to key in pivot string
                              *
 0033 10AE 44                         LDY     PTREND,U
 0036 E6   8D 00AB                    LDB     START C,PCR
 003A 1E   12                         EXG     X,Y
 003C 3A                              ABX                     No ABY available
 003D 1E   12                         EXG     X,Y
                              *
                              * Move lower pointer back a record for the loop.
                              * (Too bad there is no Subtract rB from Y)
                              *
 003F 1F   10                         TFR     X,D
 0041 E0   8D 00A1                    SUBB    RECLEN,PCR
 0045 82   00                         SBCA    #0
 0047 1F   01                         TFR     D,X
                              *
                              * Move lower bounds pointer up until a key larger than
                              * or equal to pivot key is found.
                              *
                      0049    LLOOP   EQU     *
 0049 E6   8D 0099                    LDB     KEYLEN,PCR

 004D 3A                              ABX
                              *
                              * Compare keys
                              *
 004E A7   C8                         STA     LOWER,U
 0050 E6   8D 0091                    LDB     STARTC,PCR
 0054 3A                              ABX                     Point to key
 0055 E6   8D 0088                    LDB     KEYLEN,PCR
 0059 AD   9C A8                      JSR     [SCOMPR,PCR]
 005C AE   C8                         LDX     LOWER,U
 005E 81   87                         CMPA    #LT
 0060 27   E7                         BEQ     LLOOP
                              *
                              * Move upper bounds pointer down until a key less than
                              * or equal to pivot key is found.
                              *
 0062 AE   42                         LDX     UPPER,U
                      0064    ULOOP   EQU     *
 0064 1F   10                         TFR     X,D
 0066 E0   8D 007C                    SUBB    RECLEN,PCR
 006A 82   00                         SBCA    #0
 006C 1F   01                         TFR     D,X
                              *
                              * Compare keys
                              *
 006E AF   42                         STX     UPPER,U
 0070 E6   8D 0071                    LDB     # ASTC,PCR
 0074 3A                              ABX
 0075 E6   8D 006E                    LDB     KEYLEN,PCR
 0079 AD   9C 88                      JSR     [SCOMPR,PCR]
 007C 10AE 42                         LDY     UPPER,U
 007F 81   86                         CMPA    #GT
 0081 27   E1                         BEQ     ULOOP
                              *
                              * If lower bound is still less than upper bound,
                              * swap the out-of-order pair, and continue
                              * the partitioning process.
                              *
                              * RBCLEN needed in rB whether or not the branch
                              * is taken, since EXCHAN is called in both paths.
                              *
 0083 E6   8D 005F                    LDB     RECLEN,PCR
 0087 AE   C4                         LDX     LOWER,U
 0089 AC   42                         CMPX    UPPER,U
 008B 2C   09                         BGE     RBCORS
 008D 10AE 42                         LDY     UPPER,U
 0090 AD   9D FF6E                    JSR     [EXCHAN,PCR]
 0094 20   9D                         BRA     PART
                              *
                      0096    RBCORS  EQU     *
                              *
                              * Move the pivot element to the middle.
                              *
 0096 10AE 44                         LDY     PTREND,U
 0099 AD   9D FF65                    JSR     [EXCHAN,PCR]
                              *
                              * In order to assure that the stacks will not get
                              * excessively deep, the shortest partition must be
                              * sorted first. If this is done, the stack will not
                              * get deeper than e*log2(n), where n=list length,
                              * and e is the length of the stack entry. (log2 of 1M=20)
                              *
 009D EC   C4                         LDD     LOWER,U
 009F A3   44                         SUBD    PTRBGN,U  Length of lower partition
 00A1 ED   8D 0042                    STD     LENGTH,PCR
 00A5 EC   44                         LDD     PTREND,U
 00A7 A3   C4                         SUBD    LOWER,U   Length of upper partition
 00A9 10A3 8D 0039                    CMPD    LENGTH,PCR
 00AE 2E   06                         BGT     DOLOW
                      00B0    DOHIGH  EQU     *
 00B0 8D   0A                         BSR     QH
 00B2 8D   1A                         BSR     QL
 00B4 20   2B                         BRA     QEXIT
                      00B6    DOLOW   EQU     *
 00B6 8D   16                         BSR     QL
 00B8 8D   02                         BSR     QH
 00BA 20   25                         BRA     QEXIT
                              *
                      00BC    QH      EQU     *
 00BC AE   C4                         LDX     LOWER,U   Lower bound for upper partition
 00BE E6   8D 0024                    LDB     RECLEN,PCR
 00C2 3A                              ABX
 00C3 10AE 44                         LDY     PTREND,U
                              *
                              * Call to quick requires upper limit on top of stack,
                              * PSHU X,Y leaves rX on top, so exchange.
                              *
 00C6 1E   12                         EXG     X,Y
 00C8 36   10                         PSHU    X,Y
 00CA 17   FF53                       LBSR    QUICK     Sort upper partition
 00CD 39                              RTS
                              *
                      00CE    QL      EQU     *
 00CE EC   C4                         LDD     LOWER,U   Upper bound for lower partition
 00D0 E0   8D 0012                    SUBB    RECLEN,PCR
 00D4 82   00                         SBCA    #0
 00D6 1F   01                         TFR     D,X
 00D8 10AE 46                         LDY     PTRBGN,U
 00DB 36   30                         PSHU    X,Y
 00DD 17   FF40                       LBSR    QUICK     Sort lower partition
 00E0 39                              RTS
                              *
                      00E1    QEXIT   EQU     *
 00E1 33   4B                         LEAU    8,U       Clean up user stack
 00E3 39                              RTS
                              .
```

```
SORT                                    7-19-81 TSC ASSEMBLER   PAGE   1A
Author: ML Harkness. Placed in Public Domain, 1981.

 00E4               KEYLEN  RMB     1
 00E5               STARTC  RMB     1
 00E6               RECLEN  RMB     1
 00E7               LENGTH  RMB     2
 00E9                       ENDMOD
                            END

0 ERROR(S) DETECTED
```

```
SCOMPR                          7-19-81 TSC ASSEMBLER
Author: ML Harkness. Placed in Public Domain, 1981.

 0000                       MODULE 'SCOMPR   - Vers 2.1 '
                          * * * * * * * * * * * * * * * * * *
                          * Compare two strings
                          * Inputs - rX and rY point at beginnings of two strings
                          *          rB maximum limit on comparison
                          * Output - rX,rY Preserved
                          *          rB=trash
                          *          rA= EQ for equal,
                          *              GT for string X > string Y,
                          *              LT for string X < string Y
                          * Based on 'COMPAR' K&P, p.111
                          * Except uses a maximum count in addition to EOS
                          * * * * * * * * * * * * * * * * * *
                          *
                          *REV2.1 Corrected tests for LT and GT 15JUL81 MLH
                          *
                          * * * * * * * * * * * * * * * * * *
 0000                             ENT    SCOMPR
                          *
                     0000 SCOMPR  EQU    *
 0000 36   30                     PSHU   Y,X
                     0002 COMP    EQU    *
 0002 A6   80                     LDA    ,X+
 0004 A1   A0                     CMPA   ,Y+
 0006 27   0A                     BEQ    COUNT
 0008 2E   04                     BGT    NGT
 000A 86   FF                     LDA    #LT       *REV2.1
 000C 20   11                     BRA    EXIT
                     000E NGT     EQU    *
 000E 86   86                     LDA    #GT       *REV2.1
                          *REV2.1 Removed LDA #NO from this line
 0010 20   0D                     BRA    EXIT
                          *
                     0012 COUNT   EQU    *
 0012 81   00                     CMPA   #EOS
 0014 26   04                     BNE    CNT1
 0016 86   85                     LDA    #EQ
 0018 20   05                     BRA    EXIT
                     001A CNT1    EQU    *
 001A 5A                          DECB
 001B 26   E5                     BNE    COMP
 001D 86   85                     LDA    #EQ
                     001F EXIT    EQU    *
 001F 37   30                     PULU   Y,X
 0021 39                          RTS
                          *
 0022                             END

0 ERROR(S) DETECTED

COPY                            7-19-81 TSC ASSEMBLER
Author: ML Harkness. Placed in Public Domain, 1981.

 0000                       MODULE 'COPY     - RLOAD Vers 2.0 '
                          * * * * * * * * * * * * * * * * * * * * *
                          * Specialized copy routine.
                          * Will copy the number of bytes specified
                          * in rB.
                          * The end of the source string is returned.
                          * Based on SCOPY in K&P, except uses count.
                          * Inputs - rB=byte count for copy
                          *          rX=destination address
                          *          rY=source address
                          * Outputs- rB=saved
                          *          rX=points to byte after end of dest. str.
                          *          rY=points after byte last copied from
                          *             source string.
                          *
                          * * * * * * * * * * * * * * * * * * * * *
 0000                             ENT    COPY
                          *
                     0000 COPY    EQU    *
 0000 36   04                     PSHU   B
                     0002 COPYL   EQU    *
 0002 A6   A0                     LDA    ,Y+
 0004 A7   80                     STA    ,X+
 0006 5A                          DECB
 0007 26   F9                     BNE    COPYL
                          *
                     0009 EXIT    EQU    *
 0009 37   04                     PULU   B
 000B 39                          RTS
                          *
 000C                             ENDMOD
                                  END

0 ERROR(S) DETECTED

EXCHAN                          7-19-81 TSC ASSEMBLER
Author: ML Harkness. Placed in Public Domain, 1981.

 0000                       MODULE 'EXCHAN   - RLOAD Vers 2.0 '
                          * * * * * * * * * * * * * * * * * * * * * *
                          *
                          * See EXCHAN, K&P. This routine exhanges strings
                          * or pointers, or anything other two items of
                          * equal lenght up to 256 bytes.
                          * Inputs -
                          *          rB-length of item
                          *          rX-pointer to item 1
                          *          rY-pointer to item 2
                          * Outputs- rB-saved
                          *          rX, rY-saved
                          *
                          * * * * * * * * * * * * * * * * * * * * * *
 0000                             ENT    EXCHAN
 0000                             EXT    COPY
                          *
                     0000 ITEM1   EQU    0         X displacement on U
                     0002 ITEM2   EQU    2         Y displacement on U
                          *
                     0002 EXCHAN  EQU    *
 0002 36   30                     PSHU   X,Y
 0004 30   8D 0017                LEAX   TEMPS,PCR
 0008 AD   9C F5                  JSR    [COPY,PCR] Copy 2nd item to temps
 000B 10AE C4                     LDY    ITEM1,U
 000E AE   42                     LDX    ITEM2,U
 0010 AD   9C ED                  JSR    [COPY,PCR] Copy 1st item to 2nd
 0013 31   8D 0008                LEAY   TEMPS,PCR
 0017 AE   C4                     LDX    ITEM1,U
 0019 AD   9C E4                  JSR    [COPY,PCR] Copy temps to 1st
                     001C EXIT    EQU    *
 001C 37   30                     PULU   X,Y
 001E 39                          RTS
                          *
 001F              TEMPS  RMB    256       Longest permitted string
 011F                             ENDMOD
                                  END

    0 ERROR(S) DETECTED

OSLINK - RLOAD Vers 2.0
       Length=0380
       Beginning address=0000
CD03
  FLEX
C015
  GETCHR
CD2D
  GETFIL
CD33
  SETEXT
CD39
  OUTDEC
CD3C
  OUTHEX
CD24
  PCRLF
CD18
  PSTRNG
CD18
  PUTCHR
CD3F
  RPTERR
CC22
  OUTSWC
CC23
  INPSWC
CC24
  FILOUT
CC26
  FILINP
CC1B
  FLADDR
       Length=00C5
       Beginning address=0490
04A2
  ENTPRO
EXCHAN - RLOAD Vers 2.0
       Length=011F
       Beginning address=0555
0557
  EXCHAN
EXTPRO - RLOAD Vers 2.0
       Length=0053
       Beginning address=0674
0680
  EXTPRO
EXT2 - RLOAD Vers 2.0
       Length=00A7
       Beginning address=06C7
06D1
  EXT2
GETNAM - RLOAD Vers 2.0
       Length=009D
       Beginning address=0778
0718
  GETNAM
MTEMPS - RLOAD Vers 2.1
       Length=008C
       Beginning address=07AB
07B7
  MTEMPS
PASS1 - RLOAD Vers 2.0
       Length=02BE
       Beginning address=0837
084B
  PASS1
PASS2 - RLOAD Vers 2.0
       Length=023F
       Beginnin  address=0AF5
0B08
  PASS2
RDBNRC - RLOAD Vers 2.0
       Length=01AE
       Beginning address=0D3A
0D3A
  RDBNRC
RLOAD Vers 2.0
       Length=0062
       Beginning address=0EDF
0EF3
  RLOAD
******                  Transfer address=0EF3
SCOMPR - Vers 2.1
       Length=0022
       Beginning address=0F4D
0F6D
  SCOMPR
SEARCH - RLOAD Vers 2.0
       Length=0024
       Beginning address=0F6F
CC2B
  MEMEND
  R0
  SYSFCB
D403
  FMSCLS
D406
  FMS
0100
  STACK
0100
  RDFCB
0240
  WRFCB
ABSPRO - RLOAD Vers 2.0
       Length=00EA
       Beginning address=0380
0392
  ABSPRO
COPY - RLOAD Vers 2.0
       Length=000C
       Beginning address=0834
083A
  COPY
ENTER - RLOAD Vers 2.0
       Length=004A
       Beginning address=0446
044C
  ENTER
ENTPRO - RLOAD Vers 2.0

0F71
  SEARCH
SORT - RLOAD Vers 2.0
       Length=00E9
       Beginning address=0F93
0F99
  SORT
WRBNRC - RLOAD Vers 2.0
       Length=00A3
       Beginning address=107C
1080
  WRBNRC
ZCOPY - RLOAD Vers 2.0
       Length=001B
       Beginning address=10BF
10BF
  ZCOPY
SYMTAB - RLOAD Vers 2.0
       Length=0002
       Beginning address=10DA
10DC
  SYMTAB
10DA
  SYMMEN

      **** SYMBOL TABLE ****

ABSPRO=0392      RPTERR=CD3F
COPY=0834        SCOMPR=0F4D
ENTER=044C       SEARCH=0F71
ENTPRO=04A2      SETEXT=CD33
EXCHAN=0557      SORT=0F99
EXT2=06D1        STACK=0100
EXTPRO=0680      SYMMEN=10DA
FILINP=CC26      SYMTAB=10DC
FILOUT=CC24      SYSFCB=CB40
FLADDR=CC1B      WRBNRC=1080
FLEX=CD03        WRFCB=0240
FMS=D406         ZCOPY=10BF
FMSCLS=D403
GETCHR=CD15
GETFIL=CD2D
GETNAM=0718
INPSWC=CC23
MEMEND=CC2B
MTEMPS=07B7
OUTDEC=CD39
OUTHEX=CD3C
OUTSWC=CC22
P 2S1=084B
PASS2=0B08
PCRLF=CD24
PSTRNG=CD18
PUTCHR=CD1B
RDBNRC=0D3A
RDFCB=0100
RLOAD=0EF3
```

OS-9 NOTES

From Ray Cadmus
600 W. Lee
Moberly, Mo. 65270
(816)263-1228 - Home
(816)263-6693 - Office

OS9 NOTES

The topic for the day is communications. No good reason, just that I enjoy communication projects on computers. Perhaps thats a carryover from my years as a ham radio operator experimenting with various types of communications media. Also, since I regularly work with several different types of computers, I'm constantly transferring files from one to another. The subject of computer communications can take in many and varried areas so I'll try to narrow the framework a little. Specifically we'll look at asynchronous, serial, computer to computer communications.

This is the normal mode just like a terminal uses. In fact the simplest way to explore the idea is to look at a simple dumb terminal emulator - then start enhancing it a little bit to make it somewhat more useful. In its simplest form the computer itself becomes nothing more than an expensive terminal. Then we add provision for file to file transfer. The next step is adding some kind of handshaking to insure error free data transfer with automatic retransmission of questionable data. I don't think we'll get into that this time around, but if there is enough interest expressed in that area we can pick it up in more detail at some later date.

On to basics. I'm assuming we have a computer with two serial ports, the control terminal is connected to one. The other port may be connected to another terminal, a modem, or another computer. The results should be identical. The serial port consists of two addresses which give access to four registers. For sake of discussion we'll consider a serial device at port 0 on an SS50 system. The 2 addresses used would be E000 & E001. If you write to E000 you are writing to the COMMAND register. When you read E000 you are reading the status register. Reading and writing to E001 access the respective data registers. We start out by writing $03 to the command register to initialize the port. Then we write something like $15 to the command register to set up proper word length, parity and interupt availability. Now to read a character comming in we read the status register and test for a character received bit on. If it is we can read the character from the data register. If it isn't, then we go do something else for a while. To write a character out the port we first read the status register and test for output data register empty, if it is empty we can write our character to the data register, otherwise we loop on the status test till it is empty and we can write out our character. By the way, all this assumes that we are using the normal 6850 ACIA in the serial port and that we do not have the I/O interupts turned on. That's a different ballgame.

Now that you understand all this stuff about port addresses, status bytes, bit tests etc. FORGET IT. OS9 handles all that neat stuff for you. All this garbage is handled by one of the serial device drivers normally named ACIA or something like that. As a bonus, the OS9 device drivers are interupt driven so you dont waste time waiting for I/O or dont lose a character because the computer was busy thinking about something else when a character came in.

Now - the simple dumb terminal. You have your terminal connected as the normal controlling terminal and it is using the device descriptor named TERM. Your friend has his terminal connected to the other serial port and it is using the device descriptor called T1. The idea is to check the status of TERM and if a character is ready send it to T1. Then check T1, if a character is ready, send it to TERM.

In simple form it looks like this:

```
      IF TERM READY

HEN SEND TERM CHAR TO T1.
      IF T1 READY
        HEN SEND T1 CHAR TO TERM.
                    NDLOOP
```

Now take a look at the listing called CTERM and you'll see the same thing, just dressed up a little. The one problem we run into is the fact that BASIC09 can't check device status, so we use the assembler routine INKEY to handle that. Another consideration is echo of the received or transmitted character. If you are talking with a large computer chances are it will echo everything you send so you see it as input after sending it as output. This is called full duplex. If the device on the other end does not echo everything you're working half-duplex. Our simple terminal emulator assumes full duplex.

The CTERM listing shows the use of the TMODE command from within BASIC09 to turn off the terminal echo and disable some of the control character trapping so we can send things like CTL-C etc. One other consideration is the building of the device descriptor for the output port. We should be able to make a simple assignment to a device, then use tmode to set up that device for no-pause & no-echo but I haven't had much luck with that. What I've done is modify a device descriptor with debug to kill pause and echo, then renamed it to M9 and saved it. In fact it gets loaded as part of the boot group. Perhaps a simpler way to do it would be to modify the TERM device driver source and assemble your own module.

Working from assembler does simplify some of the control functions. Take a look at the listings for CTERM in BASIC09 and for ATERM in assembler. I can't claim that either of these is the definative work in terminal emulation, but they are a start. Now maybe some of the super-coders out there will send us examples of how it SHOULD be done.

**** WARNING ****

I've had an intermittent problem for about a year with the system hanging. Nothing short of an interupt from another terminal or a reset would shake it loose. This has finally been traced to the ZENITH terminal I was using. A change to a DEC VT100 pretty much eliminated the problem. This was not just a single bad terminal, I used four or five Z19's over a period of time and had the same problem with them all. I've been in contact with ZENITH but no resolution yet. For now BEWARE THE Z19 with OS9. It generates something that OS9 cannot tolerate. **** MY WANT LIST **** GOOD SCREEN EDITOR MACRO ASSEMBLER DEBUG SYSTEM WITH DIS-ASM & TRACE OPTIMIZING C COMPILER WITH FULL K & R FEATURES LOTS OF GOOD APPLICATION SOFTWARE SO WE CAN SELL THESE SYSTEMS

By the way - for those who haven't seen it, "Computerworld" published one of it's benchmark series recently on a Smoke Signal system running OS9. Blew the socks off some pretty impressive hardware systems. Their only negative comment was reguarding the lack of application software availability.

Bye.

```
Microware OS-9 Assembler 2.1   04/01/82 11:19:02
 - OS-9 Systee Symbolic Definitions

00001         * BASIC INKEY PROC
00002         * CALLED BY - RUN INKEY(I)
00003         * ACCEPTS A PATH NO
00004         * RETURNS A CHAR OR 0
00005         *
00006  0021           TYPE    SET   SBRTN+OBJCT
00007  0081           REVS    SET   REENT+1
00008                          USE   /D0/DEFS/OS9DEFS
00009
00010         *
00011         * OS-9 System Definition File Included
00012         *
00013
00521                          opt   l
00522  0000 87CD002F           MOD   IKEND,IKNAM,TYPE,REVS,IKENT,0
00523  000D 494E4B45  IKNAM     FCS   /INKEY/
00524  0012 33E4      IKENT     LEAU  0,S           SET SP
00525  0014 AE44                LDX   4,U           GET PARM PTR
00526  0016 A684                LDA   0,X           PATH NO
00527  0018 C601                LDB   #1            GET STAT
00528  001A 103F8D              OS9   I$GSTT
00529  001D 2405                BCC   RDCHR         CHAR READY
00530  001F 8600                LDA   #0
00531  0021 A784                STA   0,X           RETURN 0
00532  0023 39        QUIT      RTS                 NO CHAR READY
00533  0024 108E0001  RDCHR     LDY   #1            READ 1 CHAR
00534  0028 103F89              OS9   I$READ
00535  002B 39                  RTS
00536  002C 6C4C98              EMOD
00537  002F           IKEND     EQU   *

00000 error(s)
00000 warning(s)
$002F 00047 program bytes generated
$01CE 00462 data bytes allocated
$134E 04942 bytes used for symbols
PROCEDURE CTERM
 0000      SHELL "LOAD TMODE"
 000E      REM TERMINAL SIMULATION
 0024      SHELL "TMODE -ECHO"
 0033      SHELL "TMODE ABORT=2" \REM ALLOW USE OF CTL-C
 0059
 005A      RUN CLS
 005E      PRINT "TERMINAL MODE - USE CONTROL-P TO ENTER SPECIAL FUNCTION MODE"
 009E      PRINT
 00A0
 00A1      DIM R,W,I,MODEM,TERMR,TERMW:BYTE
 00BC      TERMW=0 \TERMW=1
 00CA      DIM P,F:INTEGER
 00D5      DIM C0:STRING[1]
 00E1      OPEN #MODEM,"/W9"
 00ED
 00BE      LOOP
 00F0        GOSUB 10
 00F4        GOSUB 20
 00F8      ENDLOOP
 00FC
 00FB
 00FE 10   REM $$$$$ READ TERM $$$$$$$
 0118      I=TERMR
 0123      RUN INKEY(I)
 0120      IF I>0 THEN
 0139        C0=CHR$(I)
 0142        IF C0=CHR$(16) THEN
 014F          GOSUB 30
 0153          RETURN
 0155        ENDIF
 0157        PRINT #MODEM,C0;
 0162      ENDIF
 0164      RETURN
 0166
 0167 20   REM $$$$$$$$$$ READ MODEM $$$$$$$$$$
 018E      I=MODEM
 0196      RUN INKEY(I)
 01A0      IF I>0 THEN
 01AC        C0=CHR$(I)
 01B5        PRINT #TERMW,C0;
 01C0      ENDIF
 01C2      RETURN
 01C4
 01C5 30   REM SPECIAL FUNCTIONS
 01DC      SHELL "TMODE ECHO"
 01EA      INPUT "(D)ir, (R)ecelve, (S)end, (T)erminal, (Q)uit )",Z$
 0220
 0221      IF Z$="T" THEN GOSUB 30 \ RETURN
 0233      ENDIF
 0235
 0236      IF Z$="D" THEN
 0243        INPUT "DIR NAME )",Z$
 0255        CMD$="DIR "+Z$
 0264        SHELL CMD$
 0269        GOSUB 30
 0260        RETURN
 026F      ENDIF
 0271
 0272      IF Z$="R" THEN
 027F        PRINT "SORRY - NOT AVAILABLE YET!"
 0298        GOSUB 30
 02A1        RETURN
 02A3      ENDIF
 02A5
 02A6      IF Z$="S" THEN
 02B3        INPUT "ENTER FILE NAME TO TRANSFER )",F$
 02D8        OPEN #F,F$:READ
 02E4        WHILE NOT(EOF(#F)) DO
 02EF          GET #F,C0
 02F9          PRINT #MODEM,C0;
 0304        ENDWHILE
 0308        RETURN
 030A      ENDIF
 030C
 030D      IF Z$="Q" THEN
 031A        SHELL "TMODE ECHO"
 0328        END
 032A      ENDIF
 032C
 032D 30   REM MSG
 0336      SHELL "TMODE -ECHO"
 0345      PRINT "$$$ TERMINAL MODE $$$"
 035E      RETURN


PROCEDURE CTERM
SHELL "LOAD TMODE"
REM TERMINAL SIMULATION
SHELL "TMODE -ECHO"
SHELL "TMODE ABORT=2" \REM ALLOW USE OF CTL-C

RUN CLS
PRINT "TERMINAL MODE - USE CONTROL-P TO ENTER SPECIAL FUNCTION MODE"
PRINT

DIM R,W,I,MODEM,TERMR,TERMW:BYTE
TERMW=0 \TERMW=1
DIM P,F:INTEGER
DIM C0:STRING[1]
OPEN #MODEM,"/W9"

LOOP
GOSUB 10
GOSUB 20
ENDLOOP


10 REM $$$$$ READ TERM $$$$$$$
I=TERMR
RUN INKEY(I)
IF I>0 THEN
C0=CHR$(I)
IF C0=CHR$(16) THEN
GOSUB 30
RETURN
ENDIF
PRINT #MODEM,C0;
ENDIF
RETURN
```

```
20 REM $$$$$$$$$ READ MODEM $$$$$$$$$$
I=PEEK(
RUN [ACCEY(I)
IF I>0 THEN
C$=CHR$(I)
PRINT #TERMM,C$;
ENDIF
RETURN

30 REM SPECIAL FUNCTIONS
SHELL "TMODE ECHO"
INPUT "(D)ir, (R)eceive, (S)end, (T)erminal, (Q)uit )",Z$

IF Z$="T" THEN GOSUB 50 \ RETURN
ENDIF

IF Z$="D" THEN
INPUT "DIR NAME )",Z$
CMD$="DIR "+Z$
SHELL CMD$
GOSUB 50
RETURN
ENDIF

IF Z$="R" THEN
PRINT "SORRY - NOT AVAILABLE YET!"
GOSUB 50
RETURN
ENDIF

IF Z$="S" THEN
INPUT "ENTER FILE NAME TO TRANSFER )",F1$
OPEN #F,F1$:READ
WHILE NOT(EOF(#F)) DO
GET #F,C$
PRINT #MODEM,C$;
ENDWHILE
RETURN
ENDIF

IF Z$="Q" THEN
SHELL "TMODE ECHO"
END
ENDIF

50 REM MS6
SHELL "TMODE -ECHO"
PRINT "$$$ TERMINAL MODE $$$"
RETURN
```

V-DISK

A Review by E. M. (Bud) Pass, Ph.D.

With the cost of memory dropping, many 6809 FLEX users have reached the 64K byte addressing limit on their 6809 systems. Plugging another 64K board into the system and activating extended memory options on the CPU and memory boards causes FLEX to acknowledge the fact that the system has 112K bytes of memory available but does not provide any convenient means of accessing it.

Many users have converted to UNIFLEX to break the 64K barrier.

Now there is another alternative, named V-DISK. V-DISK is a facility developed by James Arbuckle. When executed, it modifies the FLEX disk driver vectors to cause extended memory beyond the first 64K to be treated as simulated drive number three (by default). This additional "virtual" drive will be regarded by most FLEX commands, utilities, and programs as simply another disk drive. Drive three will appear to have its own directory and free chain and may contain any file types which could be placed on a "real" drive. Only those programs, such as NEWDISK, which deal with the physical aspects of the drive interface hardware, or have their own disk drivers, may not be used with the "virtual" disk drive. This "virtual" drive differs from a "real" drive in the following areas:

1. Speed - It is extremely fast,
2. Reliability - It should never make a hardware error,
3. Volatility - It loses its contents without power.

The number of sectors allocated to the "virtual" disk depends upon the amount of extended memory available on the system. The user may install as much memory as the system can address. On a standard SWTPC 6809 system, there are four bits of extended memory addresses, which allows sixteen banks of 64K bytes each. However, memory addresses $EXXX and $FXXX are reserved and mirrored in every bank; so only 896K bytes of addressable memory is available in those systems. Removing the first 56K bytes, 840K bytes of extended memory is available. (With minor modifications to the CPU and I/O addressing to restrict the reserved 8K bytes to $0EXXX and $0FXXX, 960K bytes of extended memory would be available). Each sector on the "virtual" disk requires 256 bytes, just as on a "real" disk (under FLEX). During initialization, FLEX checks for the existance of each 4K block of extended memory, by attempting to modify the first two bytes of each 4K block. V-DISK avoids conflicts with FLEX by shipping the first 256 bytes of each 4K block, wasting 1/16 of the available extended address space. This allows "virtual" disk to remain intact, despite the re-booting of FLEX. If FLEX were modified to restore the changed two bytes in each 4K block of extended memory, V-DISK could easily be modified to use this space.

The first 64K extended memory board provides 203 sectors of "virtual" disk. A single 256K board (192K extended) provides 623 sectors of "virtual" disk. If the amount of exteended memory is less than 501K bytes, the "virtual" disk will be formatted with eight sectors per track; otherwise, it will be formatted with sixteen sectors per track. This is due to the FLEX restriction that track and sector numbers must fit into one byte each.

The V-DISK facility is invoked by executing the program named V-DISK, which may be followed by zero or more optional independent parameters. If "F" is entered, the virtual disk will be reformatted, regardless of its original contents; otherwise, V-DISK attempts to recover the contents of the original "virtual" disk if possible. If a digit from 0 to 3 is entered, the default drive assignment of "virtual" disk to drive three may be overridden. If "*" is entered, the "virtual" disk facility is initialized but is not accessible until VASSIGN is used to assign it a drive number. If "/" and an address is entered, it is assumed to be an ending address for an area of memory to which V-DISK may relocate itself; otherwise, it relocates itself to the end of user memory and resets MEMEND accordingly. V-DISK occupies less than 1K bytes.

V-DISK is currently delivered in several versions, as follows:

VDISK.CMD uses only the first 56K bytes in each extended memory block, and does not report soft disk errors;

VDISKREP.CMD uses only the first 56K bytes in each extended memory block, but reports soft disk errors;

VDISK.FUL uses the entire address space, and does not report soft disk errors;

VDISKREP.FUL uses the entire address space, but reports soft disk errors.

As useful as V-DISK is, several auxiliary

programs make it even more useful. The simplest is VRESTORE, which nullifies the effect of V-DISK by restoring the FLEX disk driver vectors to their values when V-DISK was called. When using USEMF or other FLEX driver vector modifiers, V-DISK should be called after their invocation, not before, if VRESTORE is to be called later.

Another auxiliary program is VCOPY, which reorganizes and copies FLEX binary files into a special format which permits very fast loading. This special format is, in actuality, a memory-image file which can be loaded sequentially into memory starting at the beginning address. VCOPY will refuse to reformat a program which would overlay itself, V-DISK, or FLEX. Programs may be copied to the "virtual" disk with the standard FLEX utility COPY in case VCOPY will not process them. V-DISK checks for the special format when a program is requested from the "virtual" disk only, it will load standard FLEX-formatted binary files from the "virtual" disk but will not load special format binary files from "real" disk.

VASSIGN is an auxiliary program, similar to ASN, which permits the "virtual" disk number to be determined, modified, assigned, or deassigned, at any point in time after the "virtual" facility has been established, but before it has been disabled by VRESTORE.

The other auxiliary programs are VLOAD and VDUMP. They facilitate high-speed loading and dumping of "virtual" disk to and from "real" disk, by copying a memory-image of extended memory to and from "real" disk. This copying is performed starting at the end of the "real" disk and progressing toward the beginning. The diskette must have enough capacity to contain all of the extended memory address range. The suggested procedure is to take a newly-formatted FLEX diskette, copy a few FLEX files to it (such as FLEX, V-DISK, VLOAD, STARTUP, etc., then LINK it if it is to be bootable), and dedicate the diskette to the use of VDUMP and VLOAD, never writing to the diskette may again except using VDUMP, The diskette may then be used as an offline storage device for backing-up and restoring "virtual" disk at high speed.

What does all this benefit the user? I performed a few timing tests using the TIME function. One of my systems is a SWTPC S-BOX running at 1 MHZ, with one 256K SMS memory board, dual QUME DT-8's connected to a DMAF2 controller and dual MPI B-52's connected to an SMS DDC-16 controller. The following results were obtained:

ASMB - 8" DISK ONLY		- ASMB VIRTUAL	- ALL VIRTUAL
130 lines	16	12	6
1000 lines	38	34	17

All times are stated in seconds. The results would have shown more pronounced differences if they had been run using the 5" disk. The assembler required about four seconds more to load from "real" disk than from "virtual" disk. It ran about twice as fast from "virtual" disk as from "real" disk. Doubling the speed of the memory and processor would double the speed of "virtual" disk, by definition.

An excellent use of "virtual" disk is for small temporary files, such as short BASIC programs being debugged, and spool files. This can prevent much "real" disk fragmentation due to multiple revisions of files; the primary danger would lie in forgetting to copy the updated file from "virtual" disk to "real" disk.

In the '68 MICRO' rating system, I would rate V-DISK as AA.

V-DISK may be obtained from the following source:
Southeastern Micro Systems, Inc.
1080 Iris Drive
Conyers, GA 30207
Telephone: 404-922-1620
The price is as follows:
V-DISK only - $99.00
V-DISK plus sources - $149.00

BIT BUCKET OOPs!!

April 8, 1982

68 MICRO JOURNAL
5900 Cassandra Smith
P.O. Box 849
Hixon, Tenn 37343

Don Williams,

I was delighted with your publication of my article on disk formatting in the April issue of 68 MICRO JOURNAL. I've been waiting for the expected deluge of disks from my offer in the article for a free source. The deluge turned out to be a trickle. Art Weller of Texas was the first to point out the problem: you did not include my address! Those who have responded found my address in previous articles. Since few people will want to type in a 37,000 character source file please publish my address as soon as possible. To spice up my offer I'm including several utility sources written by Bruno Puglia and I, including our super versatile COPY. I've enjoyed reading letters from FLEX users all over the world, today I received a delightful letter from Joe Sobleski who included several games from his 1st grade granddaughter!

Leo Taylor
18 Ridge Court West
West Haven, Conn. 06516

William Hartmann
RR 2 Box 121-1
Blue Springs, MO 64015

April 12, 1982

COMMENTS ON FLEX DISK FORMAT BY LEO TAYLOR
April 1982, 68 Micro Journal

In bringing up my system using the general version of flex I found the same things about the speed of FLEX as reported in by Mr. Taylor. One way to speed up the write operation is to use a bit map of the next available sector instead of reading the sector first. The bit map will have a bit set for every free sector.

When a sector is written the bit representing that sector is cleared and when deleting a file a bit will be set for each sector freed. When the last sector is written the bit map has to be stored on the disk. It could be stored in sector 4 and the unused portion of sector 3 (the SIR).

When writting a new sector the next free sector that is closes to the current sector is picked from the map so as to limit head movement. By using the bit map scheme the free sectors are always consolidated.

The actual file format written would not change so the read operation would not change.

In concept this would be an easy modification. However, actually it would require considerable work to find all the places in FLEX to patch. I would like to encourage anyone to try this.

2 Editors Note: We really blew it month before last; we ran a very popular subject article by Leo Taylor, on the formatting of FLEX™ disk.

WE LEFT LEO'S ADDRESS OUT!! - See what we get when we reproduce from other sources rather than let

our computer do it. Anyway here it is and my apologies; you sure let me know about that with the telephone calls and letters:

Leo Taylor
18 Ridge Court West, Apt 21C
Westhaven, Conn. 06516

Sorry 'bout that Leo and good folks.

PH.1(408) 733-6979

April 5, 1982

Ω **OmegaSoft** **Industrial Products Group**

P.O. Box 70265 Sunnyvale, CA 94086

Mr. Don Williams
68 Micro Journal
P.O. Box 849
Mixson, TN 37343

Dear Don,

In the April 1982 issue of '68' Micro Journal there appeared a review of Microware's Pascal Compiler that contains some deceiving information (un-intentional I presume). Timings were provided for Ron Anderson's prime number program for a number of different Pascal Compilers. The Times for the non-Microware compilers were based on Ron's 1MHZ system, the Helix system is 2MHZ which makes the Microware compiler's times seem to be much better than the others.

Running on a 2MHZ system with a 9600 baud CRT and using the current release (2.0) of OmegaSoft Pascal the following is the latest information :

OmegaSoft Pascal - July 1981 Prime number program :
TIME (SEC) = 30 USER BYTES = 770 TOTAL BYTES = 2790

The times for the other compilers listed in the Vail review should also be adjusted for 2MHZ operation.

Sincerely yours,

Robert Reimiller

Vail Electronics, Inc.
P.O. Box 1136
Palm Bay, Fl. 32905
April 12, 1982

Mr. Don Williams
68 Micro Journal
Mixson, Tennessee 37343

Dear Mr. Williams,

Thank you for printing our letter in the April BIT Bucket. In that letter, we included some timing data that we had compiled using Microware's Pascal, and compared that data to previously published data on other 6809 Pascal products. It was brought to our attention last week that the comparison is invalid, because the previously published benchmark tests were reportedly run on a 1 MHz processor, and our tests were run on a 2 MHz processor. The published benchmark timings, to which we compared Microware's Pascal, did not specify the processor or terminal speed that was used, so we were not aware of the discrepancy.

We also apologize to persons who tried to get in touch with us during the first week of April, when our letter and advertisement first appeared in "68 Micro Journal". An unplanned out of state trip left the phone tended only by an answering machine. We received more calls than the machine could handle, so some messages were not recorded. Because of this problem, we will continue to offer the advertised sale prices on Microware software.

Sincerely,
David Vail
Barbara J. Vail
David and Barbara Vail
Vail Electronics, Inc.
(305) 729-6363

Ronald W. Anderson
3540 Sturbridge Ct.
Ann Arbor, MI 48105
MAR. 31, 1982

Dear Don:

Just received my April '68' after talking to you this afternoon. I have one comment to make regarding the letter from Vail Electronics that ends on page 34. They have done a timing comparison of OS-9 Pascal on a 2MHZ system with my results for several other Pascals on a 1 MHZ system. Correcting their times for a 1 MHZ system yields:

Pascal	Time (sec)	User Bytes	Total Bytes
OS-9 native	54	919	6113
OS-9 pcode	112	427	10241
Lucidata pcode	158	598	3929
Dynasoft pcode	143	301	1490
OmegaSoft native	66	940	2465
TSC native	59	721	14334

That puts things in a little more accurate frame of reference. The times with OS-9 Pascal are still impressive. As we all know, timing comparisons of these compilers on a Prime Number program using only integer arithmetic, only show the relative timing in the integer arithmetic mode. I have run tests using a program with a great deal of REAL arithmetic in which the numbers come out quite differently.

Yours truly,
Ron Anderson

GIMIX INC. 1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

Press Release

GIMIX TO INTRODUCE MULTIUSER 6809 WINCHESTER SYSTEM
AT WCC BOOTH A101

GIMIX' $8998.00, 120KB 6809 system supports up to four terminals and features a 2MHz 6809 CPU, 120KB of static RAM, a 19MB (unformatted) 5¼" Winchester hard disk, a 1MB (unformatted) 5¼" floppy disk, and four serial I/O ports. Memory is expandable up to 832KB. Additional memory, mass storage capacity, and I/O for additional terminals and peripherals is optional.

The unique ability of the system to select between two operating systems, under software control, makes it an ideal system for software development.

The system price includes OS-9 level 2, a UNIX-like, multi-user, multi-tasking operating system and the OS-9 Debugger, Text Editor, and Assembler. Languages available for OS-9 include BASIC09, PASCAL, C|S COBOL, and C. The system price also includes the GMXBUG/FLEX monitor/operating system combination, a single user (56KB) operating system, capable of running any software written for FLEX.

Systems are available from stock to 30 days ARO.

GIMIX versions of OS-9* are sold under license from MICROWARE SYSTEMS CORP. GIMIX versions of FLEX* are sold under license from TECHNICAL SYSTEMS CONSULTANTS.

For further information
contact: Richard Don
GIMIX Inc.
1337 W. 37th PLACE
CHICAGO, IL 60609
(312) 927-5510

(604) 439-7005

MICRONICS RESEARCH CORP.

Microcomputers - Hardware and Software
GIMIX® Sales, Service and Support

9338 LYNN AVENUE,
ABBOTSFORD,
BRITISH COLUMBIA,
CANADA, V2S 1E2

23 March 1982

68 Micro Journal,
5900 Cassandra Smith,
Computer Publishing Center,
PO Box 849,
Mixson, IN 37343

Dear Don,

Following my letter of 15 February last, here's another listing - this time it's CHECKERS - in the series which I promised you. As far as I can tell, its only problem is that when the computer makes a multiple series of jumps it is a little on the slow side in "vanishing" the pieces over which it has jumped, but I have not yet found time to clear this. Maybe someone out there? It's partly my fault as the original version simply made the jumping piece appear directly at its destination with all jumped pieces cleared, which made it difficult to see exactly what had happened, so I changed it

to "jump","clear", "jump", "clear"....., but somehow or other, after the first jump, it seems to slow down a bit. Not that it's too much of a problem, as the opportunity for multiple jumps does not occur very often during a game. One of these days I'll get it sorted out!!

Sincerely,

A.Jones
President

```
10 REM THE GAME OF CHECKERS. AUTHOR UNKNOWN.
11 REM EXPANDED AND MODIFIED TO RUN ON CIMIX 68xx WITH 80x24 VIDEO BOARD
12 REM BY R. JONES OF MICRONICS RESEARCH CORP
13 REM 33383 LYNN AVE., ABBOTSFORD, B.C. CANADA V2S 1B2
20 GOTO 1000
23 REM: GET CURSOR X-Y CO-ORDINATES
24 PRINT #0,S(27);CHR$(11);CHR$(X9);CHR$(27);CHR$(2);CHR$(Y9);
28 RETURN
30 U=X+A1 V=Y+B: IF U<0 OR U>7 OR V<0 OR V>7 THEN RETURN
40 IF S(U,V)=0 THEN GOSUB 100: RETURN
50 IF S(U,V)<0 THEN RETURN
60 U=U+A: V=V+B: IF U<0 OR U>7 OR V<0 OR V>7 THEN RETURN
70 IF S(U,V)=0 THEN  O UR 100
80 RETURN
100 IF V=0 ABS S(X,Y)=-1 THEN Q=Q+2
110 IF ABS(Y-V)=2 THEN Q=Q+5
120 IF Y=7 THEN Q=Q-2
130 IF U=0 OR U=7 THEN Q=Q+1
140 FOR C=-1 TO 1STEP2:IF U+C<0 OR U+C>7 OR V+G<0 THEN 180
150 IF S(U+C,V+C)<0 THEN Q=Q+1:GOTO 180
160 IF U-C<0 OR U-C>7 OR V-G>7 THEN 180
170 IF S(U+C,V+ )>0 AND (B(U-C,V-G)=0 OR (U-C=X AND V= =V)) THEN Q=Q-2
180 NEXT C:IF Q>B(0)THEN B(0)=Q:B(1)=X:A(2)=Y:B(3)=U:B(4)=V
190 Q=0:RETURN
200 U=X+A:V=Y+B:IF U<0 OR U>7 OR V<0 OR V>7 THEN 220
210 IF S(U,V)=0 AND S(X-A/2,Y-B/2)>0 THEN  OSUB 100
220 RETURN
299 REM UPDAT  GAME BOARD
300 X9=0: Y9=3: GOSUB 24
305 PRINT
310 FOR Y=7 TO 0 STEP -1: PRINT Y;: FOR X=0 TO 7: PRINT TAB(X*5+4);
315 IF S(X,Y)=9 THEN PRINT " ";
320 IF S(X,Y)=0 THEN PRINT ".";
330 IF S(X,Y)=1 THEN PRINT CHR$(132);
340 IF S(X,Y)=-1 THEN PRINT CHR$(131);
350 IF S(X,Y)=-2 THEN PRINT CHR$(131);CHR$(131);
360 IF S(X,Y)=2 THEN PRINT CHR$(132);CHR$(132);
370 NEXT X: PRINT: PRINT: NEXT Y: PRINT
380 REM END OF GAME?
390 FOR L=0 TO 7
400 FOR H=0 TO 7
410 IF S(L,H)=1 OR S(L,H)=2 THEN Z=1
420 IF S(L,H)=-1 OR S(L,H)=-2 THEN T=1
430 NEXT H: NEXT L: RETURN
1000 PRINTCHR$(12); TAB(32);"CHECKERS"
1010 PRINT:PRINT:PRINT
1020 PRINT"This is the game of CHECKERS. The computer is X, and you are O."
1030 PRINT: PRINT"Squares are referred to by their co-ordinates :"
1040 PRINT"0,0 is the lower-left corner"
1050 PRINT"0,7 is the upper-left corner"
1060 PRINT 7,0 is the lower-right corner"
1070 PRINT"7,7 is the upper-right corner"
1080 PRINT: PRINT"If the computer will type 'TO' it has continue to jump."
1090 PRINT"Type two negative numbers if you can't jump any further.": PRINT
1100 REM SET UP BOARD ARRAY
1110 DIM B(4),S(7,7):G=-1:B(0)=-99
1120 DATA 1,9,1,9,0,9,-1,9,9,1,9,9,0,9,-1,9,-1,15
1130 FOR K=0 TO 7:FOR Y=0 TO 7:READ J:IF J=15 THEN 1150
1140 S(X,Y)=J:GOTO 1160
1150 RESTORE:READ S(X,Y)
1160 NEXT Y:NEXT X
1170 INPUT"Do you want to go first (Y or N)";A$
1175 PRINT CHR$(12): FOR I=0 TO 7: PRINT TAB(5*I+3); I;: NEXT I
1185 IF A$="Y" OR A$="y" THEN GOSUB 300: GOTO 1760
1190 REM CLEAR MESSAGE AREA OF DISPLAY
1200 X9=45: FOR Y9=5 TO 22: GOSUB 24
1210 PRINT"                                        ": NEXT Y9: Y9=5: GOSUB 24
1220 REM: COMPUTER'S MOVE
1230 PRINT"My move is ... ";
1235 Y8=6
1240 FOR X=0 TO 7:FOR Y=0 TO 7:IF S(X,Y)>=1 THEN 1270
1250 IF S(X,Y)=-1 THEN FOR A=-1 TO 1STEP2:B=G:GOSUB 30:NEXT A
1260 IF S(X,Y)=-2 THEN FOR A=-1 TO 1STEP2:FOR B=-1 TO 1STEP2:GOSUB 30:NEXT B:NEXT A
1270 NEXT Y: NEXT X
1440 IF B(0)=-99 GOTO 1980
1450 PRINT "FROM";B(1);",";B(2);"TO";B(3);",";B(4);:B(0)=-99
146  IF B(4)=0 THEN S(B(3),B(4))=-2 ELSE S(B(3),B(4))=S(B(1),B(2))
1470 S(B(1),B(2))=0
1480 GOSUB 300
1482 IF Z<>1 GOTO 1980
1484 IF T<>1 GOTO 1930
1486 T=0: Z=0: IF ABS(B(1)-B(3))<>2 GOTO 1260
1490 S((B(1)+B(3))/2,(B(2)+B(4))/2)=0
1500 X=B(3):Y=B(4):IF B(X,Y)=-1 THEN B=-2:FOR A=-2 TO 2STEP4:GOSUB 200
1510 IF B(X,Y)=-1 THEN FOR A=-1 TO 2STEP4:FOR B=-2 TO 2STEP4:GOSUB 200:NEXT B
1520 NEXT A: IF B(0)=-99 GOTO 1530
1525 X9=60: Y9=Y8: GO UR 24
1527 PRINT "TO";B(3);",";B(4): B(0)=-99: Y8=Y8+1: GOTO 1460
1530 GOSUB 300
1540 IF Z<>1 GOTO 1980
1550 IF T<>1 GOTO 1930
1560 T=0: Z=0
1750 REM HUMAN'S MOVE
1760 X9=45: Y9=11: GOSUB 24: GOTO 1780
1770 Y9=Y9+1: GOSUB 24
1780 INPUT"Your move ... from";E,H: X=E: Y=H
1790 Y9=Y9+1: GOSUB 24
1795 V=(E+B)/2: IF V<>INT(V) THEN PRINT"There's no such square!": GOTO 1770
1800 IF S(X,Y)=0 THEN PRINT"There's no piece there to move!": GOTO 1770
1810 IF S(X,Y)<0 THEN PRINT"That's not one of your pieces!": GOTO 1770
1820 Y9=Y9+1: GOSUB 24: INPUT"To "A,B: X=A: Y=B: Y9=Y9+1: GOSUB 24
1825 IF S(E,H)=1 AND (B-H)<0 THEN PRINT"You can't move backwards!": GOTO 1840
1830 IF S(X,Y)=0 AND ABS(A-E)<=2 AND ABS(A-E)=ABS(B-H) THEN 1850
1835 PRINT"Sorry, you can't do that!"
1840 PRINT CHR$(7)CHR$(11);:GOTO 1820
1850 I=66
1860 S(A,B)=S(E,H):S(E,H)=0:IF ABS(E-A)<>1 THEN 1910
1870 S((E+A)/2,(H+B)/2)=0
1872 Y8=Y9: GOSUB 300
1874 IF Z<>1 GOTO 1980
1876 IF T<>1 GOTO 1930
1878 T=0: Z=0: X9=45: Y9=Y8
1880 Y9=Y9+1: GOSUB 24: INPUT"TO  ";A1,B1: IF A1<0 GOTO 1910
1890 IF S(A1,B1)<>0 OR ABS(A1-A)<>2 OR ABS(B1-B)<>2 GOTO 1880
1900 E=A:H=B:A=A1:B=B1:I=I+1:I=I+13:GOTO 1860
1910 IF E=7 THEN S(A,B)=2
1920 GOSUB 300
1921 IF Z<>1 GOTO 1980
1922 IF T<>1 GOTO 1930
1923 T=0: Z=0:  OTO 1200
1925 REM WRAP UP GAME
1930 Y9=Y9+2: GOSUB 24
1940 PRINT Very good ... YOU WIN!!"
1950 Y9=Y9+1: GOSUB 24: PRINT"Another Game (Y or N)?  ";: A$=INCHR$(0)
1960 IF A$="Y" OR A$="y" GOTO 1210
1970 END
1980 Y9=Y9+2: GOSUB 24
1990 PRINT"Sorry about that! I WIN!!": GOTO 1950
```

April 9, 1982

Editor,
68 Micro Journal
P.O. Box 849
Hixson, Tennessee 37343

Dear Sir:

Here is a mini-utility for Flex 2.0. It is another version of a "pause on" command. It turns the Flex pause feature on without you having to type, "TTYSET PS=Y". I use it by placing it at the end of a command line which uses the P Command.

Sincerely,
Kenneth Drexler

Kenneth Drexler
311 Wilson Way
Larkspur, California
94939

```
                        *****PAUSE ON COMMAND*****

                        *DATE: MARCH 11, 1982

                        *COMMAND TO ENABLE THE PAUSE FEATURE
                        *IN FLEX AFTER USE OF THE P COMMAND
                        *USUALLY THE PAUSE ON COMMAND CAN BE
                        *PLACED IN THE SAME LINE AS THE P
                        *COMMAND.
                        *THE COMMAND IS CALLED BY "PSON".

                        *EQUATES
AC09                    PSFLAG  EQU     $AC09
AD03                    WARMS   EQU     $AD03

A100                            ORG     $A100
A100 20 01              PSON    BRA     PS1
A102 02                 VN      FCB     2               VERSION NUMBER

A103 86 FF              PS1     LDA A   #$FF
A105 B7 AC 09                   STA A   PSFLAG
A108 7E AD 03                   JMP     WARMS           RETURN TO FLEX
                                END     PSON

NO ERROR(S) DETECTED

   SYMBOL TABLE:

PS1     A103    PSFLAG AC09    PSON    A100    VN      A102    WARMS   AD03
```

Editor
68' MICRO JOURNAL
5900 Cassandra Smith
Computer Publishing Center
P.O. Box 849
Hixson, TN. 37343

For those readers who are using the FLEX TEXT PROCESSOR (PR), I am enclosing a macro which converts the month number in the system register to the abbreviation of the name. This conversion reduces the inherent ambiguity in dates which use numbers for both days and months.

This macro works by first loading several of the PR user registers with the month numbers. As you may note, the sequence of the numbers which are loaded is not critical so long as you can remember where each month's number is located. Next, a conditional test is performed to determine which month's abbreviation to use. This test uses two chained IF statements. There is an implicit AND between the IF statements, i.e. both IF conditions must be satisfied. This construction is necessary because PR does not recognize the logical =. In this macro the negative IF, signified by the !, is used. This form is necessary because it includes the 0 result as an operable outcome. When the te.. result is a 0, the title (.TL) associated with that test is printed. Complete month names could be substituted for the abbreviations if desired.

The listed macro is actually the header macro for setting up the top of each page. It prints the date at the upper right corner with the page number immediately
below it. Then it skips three lines before starting the text.

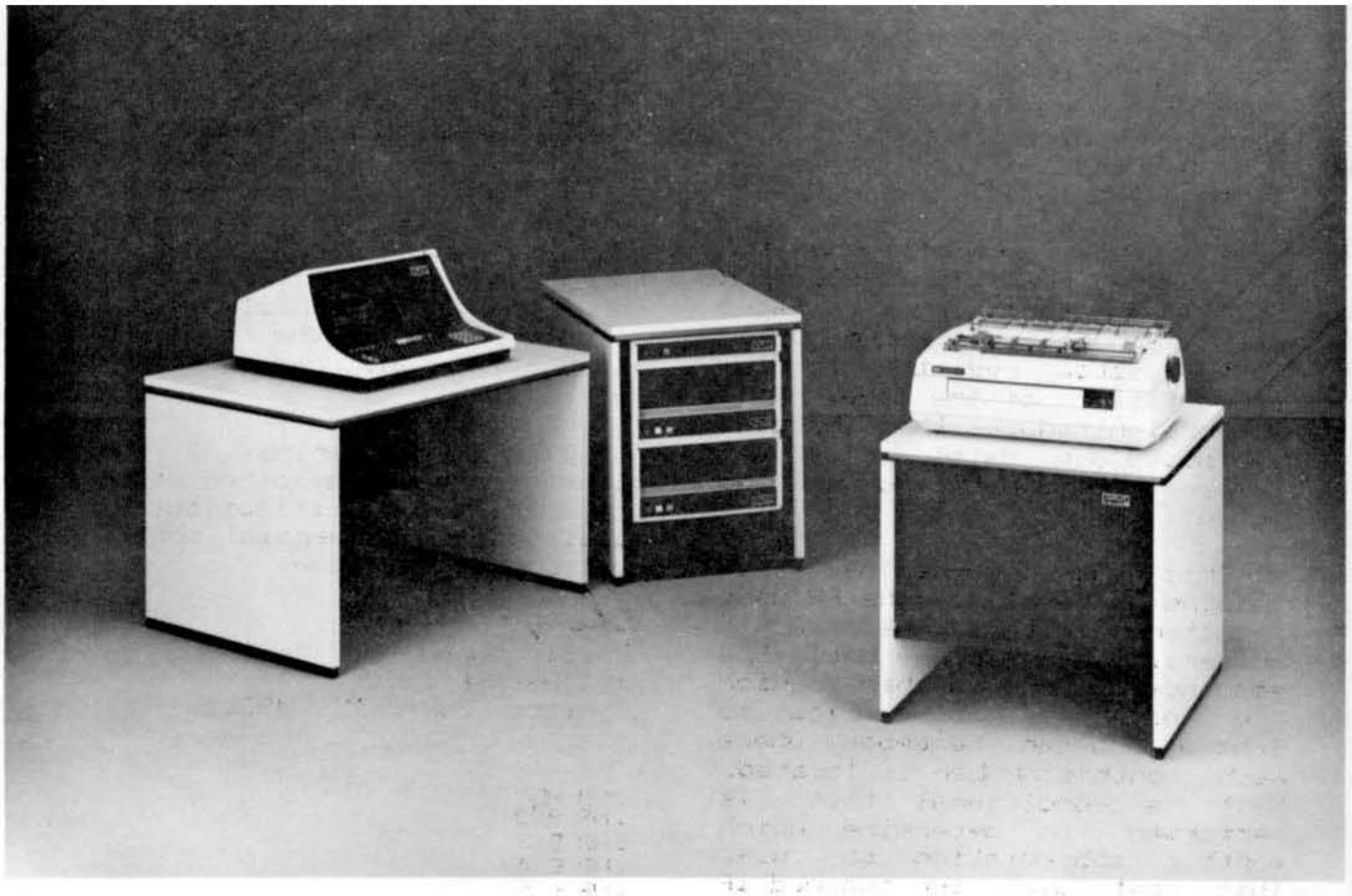
In my GIMIX 6809 system, this macro is part of the file 1.MACRO.TXT on my working disk, which is read automatically whenever PR is called. I have included the other macros of this file for reference. This file automatically formats my pages for me. If I did not want it called and read automatically, I could name the file something like 1.PAGE.TXT. Then call it at the beginning of the text file being processed, with the command .OF 1.PAGE.TXT, and read it with the command .RI.

With this offering, I would like to encourage other readers to share any clarifications and additions to commercial software which they may have.

Philip Nunn

Philip Nunn
201 Netherfield
Comstock Park, MI. 49321

```
.DM HD
.NR A 2
.NR B 3
.NR E 4
.NR F 5
.NR H 6
.NR J 7
.NR K 8
.NR Q 9
.NR R 10
.NR S 11
.NR T 12
.NR U 1
.SP 1
.IF !#M-#U .IF !#U-#M .TL '''#D-Jan-#Y'
.IF !#M-#A .IF !#A-#M .TL '''#D-Feb-#Y'
.IF !#M-#B .IF !#B-#M .TL '''#D-Mar-#Y'
.IF !#M-#E .IF !#E-#M .TL '''#D-Apr-#Y'
.IF !#M-#F .IF !#F-#M .TL '''#D-May-#Y'
.IF !#M-#H .IF !#H-#M .TL '''#D-Jun-#Y'
.IF !#M-#J .IF !#J-#M .TL '''#D-Jul-#Y'
.IF !#M-#K .IF !#K-#M .TL '''#D-Aug-#Y'
.IF !#M-#Q .IF !#Q-#M .TL '''#D-Sep-#Y'
.IF !#M-#R .IF !#R-#M .TL '''#D-Oct.-#Y'
.IF !#M-#S .IF !#S-#M .TL '''#D-Nov-#Y'
.IF !#M-#T .IF !#T-#M .TL '''#D-Dec-#Y'
.TL '''Page %'
.SP 3
.NS
.OS
..
.DM FT
.PG
..
.AT 1 HD
.AT -7 FT
.NL 4
.LM +6
.NF>>
```



THE COMPLETE BUSINESS SYSTEM

+Multiuser+Highly Expandable+Cost Effective

S+ THE CONCEPT

The S+ system is a modular computer system in which all portions of the hardware and software are designed to work together in the most efficient way possible. An S+ single user system with floppy disk storage is a competitive and cost effective entry level system. Unlike most other small computers being sold as "personal", or "small business" machines, the S+ system may be expanded to maximum capabilities using this same hardware and software. You cannot end up with a DEAD END system that cannot be expanded and whose software is not compatible with larger machines. A basic S+ system may be expanded to thirty-two users, a megabyte of main memory and hundreds of megabytes of hard disk storage by simply plugging in, or connecting the desired upgrade equipment.

TOTAL DESIGN—Hardware and Software

The S+ system is an integrated hardware and software design. The two complement and enhance each other in this system. The UniFLEX® operating system used in the S+ systems is patterned after the Bell Laboratories UNIX® operating system, one of the most admired and widely used operating systems in the world. Instead of being an afterthought, the software is part of the design of the S+ system. You can be sure that with this approach that all parts of the computer operate with maximum efficiency and cost effectiveness.

THE CENTRAL PROCESSOR

The basic S+ system is configured with 256K bytes of memory and can be expanded to more than 1 million bytes. An efficient and fast hardware memory management system is used to allocate the available memory among the users on a dynamic basis. As little as 8K bytes, or the entire memory—if needed—can be used by any individual user. This makes it possible to run very large programs on the system, but it also uses no more memory than necessary for a particular job. The increase in cost effectiveness of this system over crude and outdated bank switching arrangements is dramatic.

The central processor runs in both user and supervisor states. It can detect and reject a defective user program. It is impossible for a user program to go bad and stop the entire system, as can happen quite easily in less sophisticated systems.

Task switching is accomplished by use of a multiple map RAM memory, with sixty-four individual task maps. Each task can access from 4 to 64 K-bytes of memory. Multiple tasks may be used in programs that require more than 64K bytes of memory for execution. When a task is completed the memory is automatically released for other use.

SOFTWARE

The S+ operating system, UniFLEX® is a multiuser, multitasking operating system based on the UNIX® operating system that has been used for many years on Digital Equipment Corp. PDP-11 series minicomputers. It is considered one of the most sophisticated and "user friendly" operating systems available. Variations of UNIX® are rapidly becoming standard on mini and larger microcomputers.

A large variety of languages are available for use with the system. These include FORTRAN, COBOL, BASIC, and Pascal. Word processing packages are also available to give you full text processing capability on the system.

Applications programs are available in large quantities in many fields. This includes general business, medical, dental, veterinary, library and real estate management; plus others. Since the system is multiuser it can also be connected to cash registers to produce a point-ofsale terminal system combined with the computer. The possibilities for application of this system are endless.

THE I/O SYSTEM

The S+ system is totally interrupt driven. All terminal and printer I/O devices connect to an I/O bus separate from the main bus. Up to thirty-two separate devices may be connected to the I/O bus at any one time. If I/O activity is great enough to cause an unacceptable slowdown in system operation, a separate I/O processor can be installed in the system. This plug-in option removes all I/O handling overhead from the main processor and allows operation of up to thirty-two external devices at 9,600 baud. Without an integrated total design, as in the S+ system, it would become impractical to use a UNIX®type operating system in a situation with heavy terminal I/O activity.

DISK STORAGE

A wide range of disk storage capacity is available for the S+ system, from 2.5 M-byte floppy disks to an 80 M-byte Winchester and many sizes between. All disk controllers use direct memory access (DMA) type operations to maximize data transfer and to minimize overhead on the main processor. The Winchester disks also use intelligent controllers along with DMA transfers to preserve the performance that these type devices are capable of giving. Without this distributed intelligence the system performance would be greatly degraded. The UniFLEX®operating system is designed to work at maximum efficiency with this type disk system. The data transfer rates achieved by this combination rival those of large minicomputers.

COMMUNICATIONS

A high speed local network communications system is available to interconnect S+ systems. The VIABUS® network will allow communication between systems at data rates of over 400K baud. Such a system makes it possible to share data between local systems in an efficient and low-cost manner.

AVAILABLE SOON

Tape backup—20M-Byte in less than 15 minutes on a standard ¼ inch cartridge.

Mini-Wini—5 and 10 M-Byte Winchesters—5¼ inch package. Winchester performance, for smaller systems in a small package. UniFLEX® compatible design.

Large Capacity—190 and 340 M-Byte Winchesters, plus SMD cartridge drives.

UniFLEX is a registered trademark of Technical Systems Consultants, Inc.

UNIX is a registered trademark of Bell Labs.

VIABUS is a registered trademark of Southwest Technical Products Corporation.

DEAR DON, HERE ARE A COUPLE OF PROGRAMS THAT I HAVE FOUND QUITE USEFUL IN WORKING WITH THE FLEX OPERATING SYSTEM. THEY ARE COMPANION PROGRAMS, AND SHOULD BE PRESENTED TOGETHER. I AM ALSO INCLUDING A DISK WITH EDITOR FILES AND ASSEMBLER FILES FOR BOTH PROGRAMS, IN CASE YOU WOULD LIKE TO REGENERATE THE TEXT FOR PUBLICATION, OR COPY FOR DISTRIBUTION UNDER YOUR "68 MICRO JOURNAL PROGRAMS" SYSTEM. I BELIEVE THAT THE PROGRAMS ARE PRETTY MUCH SELF EXPLANATORY. ROGER OUSTERHOUT 511 SEYMOUR ST. OGDENSBURG, NY 13669

```
        NAM   UPSAVE.CMD

        OPT   PAG

* FOR USE WITH A SYSTEM RUNNING FLEX 2.0.

* "UPSAVE" IS A FLEX 2.0 PROGRAM TO ALLOW
* A .BIN OR .CMD FILE TO BE SAVED TO DISK
* FROM LOW MEMORY, BUT WHICH MAY RUN AT A
* DIFFERENT MEMORY LOCATION.
* FOR EXAMPLE: THE USER MIGHT USE THE COM-
* PANION PROGRAM "DOWNLOAD" TO LOAD "FLEX2.
* SYS" INTO MEMORY AT $0700 - $1FFF; USE
* MONITOR ROUTINES TO MAKE DESIRED CHANGES;
* THEN USE "UPSAVE" TO REINSTAL THE CHANGED
* "FLEX2.SYS" ONTO DISK.

* PROTOCOL FOR "UPSAVE" IS:
*  UPSAVE,<FILE SPEC>

* THE PROGRAM WILL PROMPT FOR "OFFSET".
* OFFSET IS THE HEX VALUE TO BE ADDED TO
* THE ADDRESS FROM WHICH THE PROGRAM IS
* SAVED TO DETERMINE THE HEX ADDRESS WHERE
* THE PROGRAM WILL RUN.
* THE PROGRAM WILL ALSO PROMPT FOR FOR
* ADDRESS BLOCKS TO BE SAVED.
* FOR EXAMPLE: IN THE CASE OF UPSAVING FLEX2.
* SYS: IF FLEX2 HAS BEEN LOADED WITH "DOWNLOAD"
* USING THE DEFAULT OFFSET OF $6000, IT WILL HAVE
* LOADED IN THE AREA OF $0700-$1FFF. WE WOULD
* HAVE THEN USED THE SYSTEM MONITOR TO MAKE ANY
* DESIRED CHANGES, AND NOW, WITH "UPSAVE", WHEN
* PROMPTED FOR "OFFSET", WE WOULD ENTER $A000
* ($A000+0700=A700). WHEN PROMPTED FOR BEGIN AND
* END ADDRESSES, WE WOULD ENTER THE FOLLOWING LOW
* MEMORY ADDRESSES: $0C00-0C34 (AC00-AC34), $0C49-
* 0C49 (AC49), $0C4E-0CB0 (AC4E-ACB0), $0CC0-0CC0
* (ACC0), $0CD0-0C0B (ACD0), $0CE4-0CE4 (ACE4),
* $0CFB-13C2 (ACFB-B3C2), $13FD-13FF (B3FD-B3FF),
* $0A00-0AEC (AA00-AAEC), $0B40-0B4F (AB40-AB4F),
* $0700-07FA (A700-A7FA), $1400-1400 (B400-B400),
* $1435-1E0E (B435-BE0E), $1E00-1EA2 (BE00-BEA2),
* $1E03-1FB0 (BE03-BF00). ON THE NEXT PROMPT FOR
* "BIGIN ADDRESS", ENTER A CARRAIGE RETURN ONLY.
* WHEN PROMPTED FOR "TRANSFER ADDRESS", ENTER
* A000. YOU WILL NOW HAVE "SAVED" FLEX2 IN IT'S
* ORIGINAL DISK FORMAT WITH YOUR CHANGES INSTALLED.
* THE PROGRAM WILL ALSO PROMPT FOR "TRANSFER
* ADDRESS".
* ANSWERING ANY PROMPT WITH ONLY A "CR" WILL
* ASSUME "NO VALUE".

* WRITTEN BY ROGER OUSTERHOUT, 14 FEB 82.

FCB     EQU   $AB40
SETEXT  EQU   $AD33

UPSAVE.CMD                          3-23-82  ISC ASSEMBLER

AD2D            GETFIL  EQU   $AD2D
AD03            WARMS   EQU   $AD03
AD1E            PSTRNG  EQU   $AD1E
B406            FMS     EQU   $B406
B403            FMSCLS  EQU   $B403
AD3F            RPTERR  EQU   $AD3F     REPORT ERROR
AD09            INCH    EQU   $AD09     INPUT CHARACTER FROM TERM

A100                    ORG   $A100

A100 20 0F      UPSAV   BRA   UPSAV1
A102 01         VN      FCB   1         VERSION NUMBER
A103            BEGA    RMB   2         BEGIN ADR STORAGE
A105            ENDA    RMB   2         END ADR STORAGE
A107            OFFSET  RMB   2         OFFSET VALUE
A109            TEMP2   RMB   2         TEMPORARY STORAGE
A10B            TEMP3   RMB   2         TEMPORARY STORAGE
A10D            SIZE    RMB   2         BLOCK SIZE
A10F            RUNADD  RMB   2         RUN ADDRESS

A111 CE A8 40   UPSAV1  LDX   #FCB      POINT TO FCB
A114 BD AD 2D           JSR   GETFIL    GET FILE SPEC
A117 25 49              BCS   UPS9
A119 86 02              LDA A 02        OPEN FOR WRITE CODE
A11B A7 00              STA A 0,X       SET IT
A11D 4F                 CLR A           SET .BIN EXTENSION DEFAULT
A11E BD AD 33           JSR   SETEXT    SET EXTENSION
A121 BD B4 06           JSR   FMS       DO OPEN
A124 26 3C              BNE   UPS9      ERROR CHECK
A126 CE AB 40           LDX   #FCB
A129 86 FF              LDA A #$FF      SET FOR BINARY WRITE
A12B A7 3B              STA A $3B,X     SET COMPRESSION FLAG
A12D 4F                 CLR A           WRITE NEXT BYTE CODE
A12E A7 00              STA A 0,X       SET IT
A130 CE A2 6C           LDX   #OFSTMG   /ENTER OFFSET VALUE (4 HEX) ? /
A133 BD AD 1E           JSR   PSTRNG
A136 BD A1 99           JSR   BADDR     GET HEX VALUE
A139 24 03              BCC   DOFF
A13B CE 00 00           LDX   #0        DEFAULT VALUE
A13E FF A1 07   DOFF    STX   OFFSET
A141 CE A2 A7   DOBLK   LDX   #BEGMSG   /ENTER BLOCK BEGIN ADR ? /
A144 BD AD 1E           JSR   PSTRNG    PRINT IT
A147 BD A1 99           JSR   BADDR     BUILD ADR IN X
A14A 25 19              BCS   DOTRAN
A14C FF A1 03           STX   BEGA
A14F CE A2 CB           LDX   #ENDMSG   /ENTER BLOCK END ADR ? /
A152 BD AD 1E           JSR   PSTRNG
A155 BD A1 99           JSR   BADDR
A158 25 0B              BCS   DOTRAN
A15A FF A1 05           STX   ENDA
A15D BD A1 E0           JSR   SAVBLK    SAVE BLOCK TO DISK
A160 20 DF              BRA   DOBLK
A162 7E A2 63   UPS9    JMP   DOERR
A165 CE A2 E7   DOTRAN  LDX   #TRNMSG   /ENTER TRANSFER ADR ? /
A168 BD AD 1E           JSR   PSTRNG
A16B BD A1 99           JSR   BADDR
A16E 25 14              BCS   SAVB
A170 FF A1 03           STX   BEGA
A173 CE A8 40   SAVTRN  LDX   #FCB      POINT TO FCB
A176 86 16              LDA A #$16      TRANSFER DELIMETER
A178 BD 19              BSR   STORE
A17A 86 A1 03           LDA A BEGA      TRANSFER MSBYTE
A17D BD 14              BSR   STORE
A17F B6 A1 04           LDA A BEGA+1    LSBYTE
A182 8D 0F              BSR   STORE
A184 CE A8 40   SAVB    LDX   #FCB      CLOSE FILE
A187 86 04              LDA A #4        CLOSE FILE CODE
A189 A7 00              STA A 0,X       SET IT
A18B BD B4 06           JSR   FMS       CLOSE FILE
A18E 26 D2              BNE   UPS9
A190 7E AD 03           JMP   WARMS
A193 BD B4 06   STORE   JSR   FMS       SEND BYTE
A196 26 CA              BNE   UPS9
A198 39                 RTS

A199 8F A1 0B   BADDR  STS   TEMP3      SAVE STACK
A19C BD 0D             BSR   BYTE       READ 2 FRAMES
A19E B7 A1 09          STA A TEMP2
A1A1 BD 08             BSR   BYTE
A1A3 B7 A1 0A          STA A TEMP2+1
A1A6 FE A1 09          LDX   TEMP2
```

```
A1A9 0C             CLC             SIGNAL GOOD HEX
A1AA 39             RTS
A1AB 8D 09    BYTE  BSR   INHEI     GET HEX CHAR
A1AD 48       BYTEI ASL A
A1AE 48             ASL A
A1AF 48             ASL A
A1B0 48             ASL A
A1B1 16             TAB
A1B2 8D 02          BSR   INHEX
A1B4 1B             ABA
A1B5 39             RTS
A1B6 BD AD 09 INHEX JSR   INCH      GET CHARACTER
A1B9 81 09          CMP A #$0D      IS IT CR ?
A1BB 27 13          BEQ   NOHEX
A1BD 80 30    INHEX1 SUB A #$30
A1BF 2B 14          BMI   NOTHEX
A1C1 81 09          CMP A #9
A1C3 2F 0A          BLE   INRTS
A1C5 81 11          CMP A #$11
A1C7 2B 0C          BMI   NOTHEX
A1C9 81 16          CMP A #$16
A1CB 2E 08          BGT   NOTHEX
A1CD 80 07          SUB A #7
A1CF 39       INRTS RTS

A1D0 BE A1 0B NOHEX LDS   TEMP3     RETRIEVE GOOD STACK
A1D3 0D             SEC             SIGNAL NO NUMBER
A1D4 39             RTS

A1D5 CE A2 BA NOTHEX LDX  #NHXMSG   /NOT HEX, REENTER 4 HEX ? /
A1D8 BD AD 1E       JSR   PSTRNG
A1DB BE A1 0B       LDS   TEMP3
A1DE 20 BC          BRA   BADDR+3

A1E0 B6 A1 05 SAVBLK LDA A ENDA     COMPUTE BLOCK SIZE
A1E3 F6 A1 06       LDA B ENDA+1
A1E6 F0 A1 04       SUB B BEGA+1
A1E9 B2 A1 03       SBC A BEGA
A1EC B7 A1 0D       STA A SIZE      BLOCK SIZE MSBYT
A1EF F7 A1 0E       STA B SIZE+1
A1F2 FE A1 0D       LDX   SIZE
A1F5 08             INX             ADJUST IT
A1F6 FF A1 0D       STX   SIZE
A1F9 B6 A1 07       LDA A OFFSET    COMPUTE RUN ADR
A1FC F6 A1 08       LDA B OFFSET+1
A1FF F0 A1 04       ADD B BEGA+1
A202 B9 A1 03       ADC A BEGA
A205 B7 A1 0F       STA A RUNADD    RUN ADR STORAGE
A208 F7 A1 10       STA B RUNADD+1
A20B B6 A1 0D DOREC LDA A SIZE      DETERMINE RECORD SIZE
A20E F6 A1 0E       LDA B SIZE+1
A211 C0 C4          SUB B #$C4      SEE IF $C4 TOO BIG
A213 B2 00          SBC A #0
A215 25 04          BCS   TOBIG
A217 B7 A1 0D       STA A SIZE
A21A F7 A1 0E       STA B SIZE+1
A21D C6 C4          LDA B #$C4
A21F 20 05          BRA   SAVREC
A221 F6 A1 0E TOBIG LDA B SIZE+1
A224 27 3C          BEQ   BXDON
A226 B6 02    SAVREC LDA A #2       RECORD START INDICATOR
A228 CE A8 40       LDX   #FCB
A22B BD B4 06       JSR   FMS
A22E 26 33          BNE   DOERR
A230 B6 A1 0F       LDA A RUNADD    SAVE RUN ADR
A233 BD B4 06       JSR   FMS
A236 26 28          BNE   DOERR
A238 B6 A1 10       LDA A RUNADD+1
A23B BD B4 06       JSR   FMS
A23E 26 23          BNE   DOERR
A240 17             TBA             SAVE RECORD BYTE COUNT
A241 BD B4 06       JSR   FMS
A244 26 1D          BNE   DOERR
A246 FE A1 03 SAV7  LDX   BEGA      SAVE RECORD DATA
A249 A6 00          LDA A 0,X
A24B CE A8 40       LDX   #FCB
A24E BD B4 06       JSR   FMS
A251 26 10          BNE   DOERR
A253 FE A1 03       LDX   BEGA
A256 08             INX
A257 FF A1 03       STX   BEGA
A25A 5A             DEC B           BYTE COUNTER
A25B 27 02          BEQ   SAV6      DO ANOTHER RECORD
A25D 20 E7          BRA   SAV7

A25F 7E A1 E0 SAV6  JMP   SAVBLK

A262 39       BXDON RTS

A263 BD AD 3F DOERR JSR   RPTERR
A266 BD B4 03       JSR   FMSCLS
A269 7E AD 03       JMP   WARMS

A26C 45       OFSTMG FCC  /ENTER OFFSET VALUE (4 HEX) ? /
A289 04             FCB   4
A28A 4E       NHXMSG FCC  /NOT HEX. RE-ENTER (4 HEX) ? /
A2A6 04             FCB   4
A2A7 45       BEGMSG FCC  /ENTER BLOCK BEGIN ADR (4 HEX) ? /
A2C7 04             FCB   4
A2C8 45       ENDMSG FCC  /ENTER BLOCK END ADR (4 HEX) ? /
A2E6 04             FCB   4
A2E7 45       TRNMSG FCC  /ENTER TRANSFER ADR (4 HEX) ? /
A304 04             FCB   4

                    END   UPSAV
```

NO ERROR(S) DETECTED

```
                    NAM   DOWNLOAD.CMD

                    OPT   PAG

                * 'DOWNLOAD' IS A FLEX 2.0 PROGRAM TO ALLOW
                * A .BIN OR .CMD FILE TO BE LOADED IN MEMORY
                * AT AN ADDRESS OTHER THAN IT'S RUNNING LOCATION.
                * IT IS VALUABLE IN ALLOWING THE OPERATOR TO
                * TRANSFER FROM DISK TO MEMORY A BINARY ENCODED
                * FILE FOR EXAMINATION OR MAKING CHANGES TO A
                * FILE WHICH IS WRITTEN TO RUN IN THE COMMAND
                * FILE AREA, OR TO TRANSFER TO RAM A PROGRAM
                * DESTINED TO RUN IN EPROM (ALLOWS USER TO USE
                * EXISTING EPROM BURNER PROGRAMS).

                * THE COMPANION PROGRAM WHICH PERFORMS THE
                * OPPOSITE FUNCTION IS "UPSAVE".

                * PROTOCAOL FOR "DOWNLOAD" IS:
                *  DOWNLOAD,<FILE SPEC>[,HEX VALUE ]
                *
                * "HEX VALUE" IS OPTIONAL, AND DETERMINES
                * THE VALUE TO BE ADDED TO THE NORMAL LOAD
                * ADDRESS OF THE FILE TO DETERMINE THE NEW
                * LOAD ADDRESS.  DEFAULT VALUE IS $6000,
                * SO THAT A .CMD FILE WHICH NORMALLY
                * LOADS AT $A100, FOR EXAMPLE, DEFAULTED,
                * WOULD LOAD AT $0100 ($A100+6000=$0100).
                * IF THE OBJECT FILE IS LOADED TO DISCONTINUOUS
                * RAM, THE SEPARATIONS WILL REMAIN.

AD30        LOAD  EQU   $AD30
AB40        FCB   EQU   $AB40
AD33        SETEXT EQU  $AD33
AD2D        GETFIL EQU  $AD2D
AD03        WARMS EQU   $AD03
AD1E        PSTRNG EQU  $AD1E
AD42        GETHEX EQU  $AD42
B406        FMS   EQU   $B406
B403        FMSCLS EQU  $B403
AC11        LASTRN EQU  $AC11
AC10        LDOFST EQU  $AC10     LOAD OFFSET ADDRESS
AD3F        RPTERR EQU  $AD3F     REPORT ERROR
```

```
A100                    ORG     $A100

A100 20 01      DWNLD   BRA     DWNLD1
A102 01         VN      FCB     1           VERSION NUMBER
A103 CE A8 40   DWNLD1  LDX     #FCB        POINT TO FCB
A106 BD AD 2D           JSR     GETFIL      GET FILE SPEC
A109 25 45              BCS     DWN9
A10B B6 01              LDA A   #1          OPEN FOR READ CODE
A10D A7 00              STA A   0,X         SET IT
A10F 4F                 CLR A               SET .BIN EXT DEFAULT
A110 BD AD 33           JSR     SETEXT      SET EXTENSION
A113 B6 AC 11           LDA A   LASTRM      GET LAST TERMINATOR
A116 81 2C              CMP A   #$2C        IS IT COMMA ?
A118 27 0F              BEQ     DWN1
A11A 81 20              CMP A   #$20        IS IT SPACE ?
A11C 27 0B              BEQ     DWN1
A11E 81 3A              CMP A   #$3A        IS IT COLON ?
A120 27 11              BEQ     DWN2
A122 81 0D              CMP A   #$0D
A124 27 0D              BEQ     DWN2
A126 7E A1 59           JMP     SYNERR
A129 BD AD 42   DWN1    JSR     GETHEX      GET HEX ADDRESS
A12C 25 33              BCS     NUMERR
A12E FF AC 1B   DWN4    STX     LDOFST      SET LOAD OFFSET VALUE
A131 20 05              BRA     DWN3
A133 CE 60 00   DWN2    LDX     #$6000      DEFAULT VALUE
A136 20 F6              BRA     DWN4
A138 CE A8 40   DWN3    LDX     #FCB
A13B BD B4 B6           JSR     FMS         DO OPEN
A13E 26 10              BNE     DWN9
A140 B6 FF              LDA A   #$FF        SET FOR BINARY READ
A142 A7 38              STA A   $38,X       SET COMPRESSION FLAG
A144 BD AD 38           JSR     LOAD        LOAD THE FILE
A147 CE 00 00           LDX     #0
A14A FF AC 1B           STX     LDOFST      CLEAR OFFSET VALUE
A14D 7E AD 03           JMP     WARMS
A150 BD AD 3F   DWN9    JSR     RPTERR      REPORT ERROR
A153 BD B4 03   DWN5    JSR     FMSCLS      CLOSE ALL FILES
A156 7E AD 03           JMP     WARMS
A159 CE A1 69   SYNERR  LDX     #SYNMSG     /SYNTAX ERROR, REENTER COMMAND/
A15C BD AD 1E           JSR     PSTRNG
A15F 20 F2              BRA     DWN5
A161 CE A1 87   NUMERR  LDX     #NUMMSG     /HEX NUMBER ERROR, REENTER COMMAND/
A164 BD AD 1E           JSR     PSTRNG
A167 20 EA              BRA     DWN5
A169 53         SYNMSG  FCC     /SYNTAX ERROR, REENTER COMMAND/
A186 04                 FCB     4
A187 48         NUMMSG  FCC     /HEX NUMBER ERROR, REENTER COMMAND/
A1A8 04                 FCB     4
                        END     DWNLD

NO ERROR(S) DETECTED
```

Electronic Specialists, Inc.
171 South Main Street, Natick, Mass. 01760
(617) 655-1532

NEW PRODUCT RELEASE

FOR IMMEDIATE RELEASE

FOR MORE INFORMATION: FRANK STIFTER

DIRECT PLUG ISOLATORS

ELECTRONIC SPECIALISTS expands their patented ISOLATOR Line to include units that plug directly into the wall socket. Designed for installations that do not require extension cords, the Direct Plug ISOLATORS provide the same equipment interaction isolation and power line protection as their popular line cord ISOLATOR series. A convenient retention screw prevents accidental withdrawal from the wall socket.

Direct Plug ISOLATORS can accommodate a total 1875 watt load, with up to 1000 watts per socket. A high capacity Spike/Surge Suppressor is designed into each unit.

Direct Plug SUPER ISOLATOR (Model DP-SISO32) provides 2 super-isolated channels for $96.95.

Electronic Specialists, Inc., 171 S. Main St., Box 389
Natick, Massachusetts 01760 Phone: (617) 655-1532

NEWSRELEASE

FOR IMMEDIATE RELEASE

COMPARATIVE STUDY AND REPORT ON THE CHIEFTAIN SERIES AVAILABLE FROM THE MANUFACTURER

WESTLAKE VILLAGE, CA...SMOKE SIGNAL, manufacturer of the CHIEFTAIN (tm) Series of Business Computers based on the 16 bit 6809 processor, has announced the availability of a report based on a series of benchmarks that were performed on the manufacturer's computer systems by the Association of Computer Users and the Business Research Division of the University of Colorado.

The Association of Computer Users is a world-wide, non-profit organization, whose many facets include a monthly magazine as well as responsibility for the unbiased Benchmark Reports and associated articles in major trade publications. The Association of Computer Users, in collaboration with the University of Colorado, perform three separate series of benchmark tests on computers which are divided into price categories. Smoke Signal's CHIEFTAIN was tested in the Under-$25,000 category.

In a new evaluation from ACU and the University of Colorado, Business Research Division, the Smoke Signal CHIEFTAIN was found to be "one of the fastest tested in this series." The CHIEFTAIN was also one of the only systems tested with multi-user, multi-tasking capability in this price range, and with features comparable to those of advanced systems.

The Benchmark Report also discusses the reaction of end-users. According to ACU, "Users remarked that the 'OS-9 Operating System and BASIC09 High-Level Language were excellent'." Another user went on to say, "I bought it (CHIEFTAIN) because it was obvious that it was the best around. The best feature is the reliability, it just sits there and works!!"

Ric Hammond, President of Smoke Signal, said, "We felt that ACU and the University of Colorado were very thorough in their analysis of the CHIEFTAIN. We were exceptionally pleased with the results of the tests and the overall reactions of the unbiased and independent parties involved in the evaluation--especially the end-users."

The Benchmark Report on the system covered five main testing categories: CPU and I/O intensive for speed; a Scientific and Engineering Test which exercises compute times and speed; A New Product Planning Problem application; an Accounts Receivable generation application; and an 'Ease of Use' Test.

All of these tests are implemented the same regardless of the manufacturer and the differences between the computer systems within that particular range; in this case, the Under $25,000 category. The results of the tests are described in the 24-page report written by ACU and the University of Colorado.

Smoke Signal's CHIEFTAIN 9822 (double-side, double-density 8-inch floppy based system) and the CHIEFTAIN 9810 (a 10 Megabyte 8-inch Winchester and an 8-inch double side, double density floppy or 20 Megabyte Tape Streamer) were tested running the new UNIX compsrable multi-user, multi-

tasking, operating system, OS-9. BASIC09, the incremental compiler BASIC for the OS-9 operating system was also tested.

The CHIEFTAIN Series Benchmark Report also includes detailed descriptions of the hardware components, the software components, the support components, and the pricing of each system. A highlight of the report is the interviews with users of the systems that ACU and the University conducted, and comments by the users are contained throughout the entire report. Also included are comments from ACU and the Business Research Division of the University of Colorado; what they thought of the CHIEFTAIN, and an extensive analysis of their overall observations of the performance and impression of the CHIEFTAIN Series.

The CHIEFTAIN Series of computers range from 5 1/4-inch floppies, to 8-inch floppies, to 5 1/4-inch Winchesters, to 8-inch Winchesters. The CHIEFTAIN Winchester based systems range from 4 to 70 Megabyte configurations. A 20 Megabyte 1/4-inch tape streamer for backing up the larger storage hard disk computers is available also. RAM capacity for the standard CHIEFTAIN is 64K (and upgradeable to 1 Megabyte RAM), all systems come standard with Smoke Signal's own single-user operating system - DOS69D - and two serial I/O ports.

For a copy of the Association of Computer Users BENCHMARK REPORT on SMOKE SIGNAL'S CHIEFTAIN Series of Business Computers, please write or call Smoke Signal directly.

Smoke Signal will be exhibiting the CHIEFTAIN Series at NCC '82 booths #2922-2923, June 7-10, 1982, at the Astrodarena in Houston, Texas.

PLEASE CONTACT:

Deborah Conrad, Manager, Dealer Sales and Support
Jim Allday, National Sales Manager

SMOKE SIGNAL 31336 Via Colinas, Westlake Village, CA 91361 • Tel (713) 889-9340

Smoke Signal Chieftains Set New Standards For Speed and Cost-Effectiveness

	C-3 Accounts Receivable Time	Current Price
CHIEFTAIN 9822	1:40.7	$ 8,149
CHIEFTAIN 98W10	:35.2	12,149
PERTEC PC 2000	6:04.3	12,470
NORTH STAR HORIZON	1:57.7	6,911
CROMEMCO SYSTEM TWO	2:48.0	9,275
TEXAS INSTRUMENTS MODEL 771	3:38.1	12,100
VECTOR GRAPHIC SYSTEM B	5:56.5	8,995
DECSTATION 78	5:04.8*	10,495
RADIO SHACK TRS-80 MODEL II	3:38.6	7,609
APPLE II+	8:17.4	4,330
DIGITAL MICROSYSTEMS DSC-2	3:28.8	9,015
OHIO SCIENTIFIC C3-A	15:49.3	10,940
ALPHA MICRO AM-1011	3:25.3	15,605
DATA GENERAL CS/10	2:40.3**	13,400
ALTOS ACS8000-15	10:41.5	9,875
DYNABYTE 5300	11:01.5	8,535
IBM 5120	4:18.2	13,705

*Result includes both compile and run time.
**Time of 2:40.3 was obtained using hard disk system.

Prices are list prices of configurations tested including CRT and Printer.

March 25, 1982

'68' Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Dear Mr. Williams:

First off, I want to say how pleased I am to hear from several readers in response to my last letter; and to thank again those who took the time to show interest and to share some of what they're doing.

Here is a routine called "GETNUM" that functions similarly to FLEX's "GETHEX" routine except that this one can be used to input numbers in binary, octal, decimal, hexidecimal, or any base from 2 to 16 (try out base 5 or maybe 11(). In fact, it is not limited to this range, but more characters would have to be added (G,H,etc.) for bases beyond 16. This program uses the stack for a work area and also demonstrates how handy is the hardware multiply of the 6809. Don't look at the address of NXTCH in the listing as this is it's address in my own system. Any other questions should be answered in the listing.

Sincerely,

Randy L. Kron

Randy L. Kron
Rt. 2
Kalona, Iowa 52247

```
6809 RESIDENT ASSEMBLER V1.0  05/25/82 01:10:45 PM
PAGE 001  GETNUM

0001                            NAM   GETNUM
0002    E0 2A                   NXTCH   EQU   $E02A
0003 0000                       *
0004 0000                       * STACK DEPTH
0005 0000                       *
0006    00 00                   MPLIER  EQU   0
0007    00 01                   CURVAL  EQU   1
0008    00 02                   PWRD    EQU   2
0009 0000                       *
0010 0000                       * GETNUM CAN RETRIEVE A NUMBER IN ANY BASE
0011 0000                       * FROM BASE 2 TO BASE 16 FROM THE FLEX LINE
0012 0000                       * BUFFER. THE BINARY VALUE IS RETURNED
0013 0000                       * IN THE X REGISTER, AND B AND CARRY ARE
0014 0000                       * SET CONSISTENT WITH TSC'S GETHEX ROUTINE
0015 0000                       *
0016 0000                       * THIS ROUTINE WILL NOT ALLOW OVERFLOW AS
0017 0000                       * WILL TSC'S GETHEX. WHILE OVERFLOW IS OK FOR
0018 0000                       * HEX AND BASES THAT ARE POWERS OF 2, FOR
0019 0000                       * OTHER BASES THE RETURNED VALUE WILL HAVE
0020 0000                       * NO RELEVANCE TO WHAT WAS ENTERED
0021 0000                       *
0022 0000                       * INHEX AND INDEC ARE EXAMPLES OF USE
0023 0000                       *
0024 0000 C6 10                 INHEX   LDB   #16       LOAD BASE NUMBER IN B
0025 0002 20 02                         BRA   GETNUM
0026 0004 C6 0A                 INDEC   LDB   #10
0027 0006                       *
0028 0006                       * B CONTAINS NUMBER BASE ON ENTRY
0029 0006                       * ON EXIT:
0030 0006                       *   B = NUMBER OF DIGITS FOUND
0031 0006                       *   X = BINARY VALUE INPUT
0032 0006                       *   U,Y UNCHANGED
0033 0006                       *   A IS CHANGED
0034 0006                       *   CARRY SET IF NUMBER IN X IS INVALID
0035 0006                       *   CARRY CLEAR IF NUMBER IS GOOD
0036 0006                       *
0037 0006 34 40                 GETNUM  PSHS  U          SAVE USER'S U
0038 0008 6F E2                         CLR   ,-S        CLEAR THE WORK AREA
0039 000A 6F E2                         CLR   ,-S
0040 000C 6F E2                         CLR   ,-S
0041 000E E7 E2                         STB   ,-S        MULTIPLIER
0042 0010 1F 43                         TFR   S,U
0043 0012 BD E0 2A              GETDLP  JSR   NXTCH      PUNCTUATION IS PROPER TERMINATO
0044 0015 25 2B                         BCS   RETNUM     NOT A GOOD DIGIT
0045 0017 A0 2E                         SUBA  ENVT
0046 0019 25 1F                         BCS   BADNUM     NOT A GOOD DIGIT
0047 001B E6 C4                         LDB   MPLIER,U
0048 001D 34 02                         PSHS  A
0049 001F A6 43                         LDA   PROD+1,U
0050 0021 3D                            MUL              LSB OF PROD TIMES MULTIPLIER
0051 0022 EB E0                         ADDB  ,S+        ADD IN NEW DIGIT
0052 0024 89 00                         ADCA  #0
0053 0026 E7 43                         STB   PROD+1,U
0054 0028 34 02                         PSHS  A          CARRY FROM LAST OPERATION
0055 002A A6 42                         LDA   PROD,U
0056 002C E6 C4                         LDB   MPLIER,U
```

March 22,1982

68 Micro Journal
5900 Cassandra Smith
Computer Publishing Center
P.O. Box 849
Hixton, TN. 37343

Enclosed is a small program which operates as a Utility for the 6800 FLEX 2 DOS. The Utility was written as an aid to making EPROM's for Monitor use. Using the standard FLEX Syntax; BIMOVE.<file spec>, the Utility will prompt you with the question, "MOVE BINARY FILE TO?". Supply the desired HEX address. [XXXX) and the file will be placed anywhere in memory.

This program takes care of the problem of the first four bytes in the FLEX 2, 256 byte disk sector which is disk related information. If this program is to be used with MINI-FLEX a different type of counter would be needed for the 128 byte sector.

Yours truly,

William M. Hall
6232 Woodcrest Ln.
Dallas, Tx. 75214

```
                    * BIMOVE *
                    * BINARY FILE MOVE UTILITY *

                    * DOS EQUATES *

A07F                STACK   EQU     $A07F
A003                W4RMS   EQU     $A003       DOS WARM START ENTRY
AD15                GETCHR  EQU     $AD15       INPUT CHAR ROUTINE
AD18                PUTDIR  EQU     $AD18       OUTPUT CHAR ROUTINE
AD20                GETFIL  EQU     $AD20       GET FILE SPECIFICATIONS
AD1E                PSTRING EQU     $AD1E       PRINT DATA STRING
AD24                PCRLF   EQU     $AD24       PRINT CR & LF
AD33                SETEX1  EQU     $AD33       SET DEFAULT NAME EXT.
AD3F                RPTERR  EQU     $AD3F       REPORT DISK ERROR

                    * FMS EQUATES *

8406                FMS     EQU     $8406
8403                FMSCLS  EQU     $8403

                    * SYSTEM EQUATES *

A840                FCB     EQU     $A840       SYSTEM FCB


                    * GET MEMORY BASE ADDRESS *

A100                        ORG     $A1B0

A100 20 01                  BRA     GO
A102 01                     NOP                 VERSION 1
A103 CE A1 88       GO      LDX     #BEGIN      GET MESSAGE
A106 BD AD 1E               JSR     PSTRING     OUTPUT MESSAGE
A109 BD A1 9F               JSR     INPUT       GET ADDRESS
A10C FF A1 08               STX     XTEMP1      STORE BASE ADDRESS
A10F BD AD 24               JSR     PCRLF       PRINT CR & LF
A112 20 00                  BRA     LIST1       OPEN FILE

                    * OPEN DISK FILE *

A114 CE A8 40       LIST1   LDX     #FCB        POINT TO FCB
A117 BD AD 20               JSR     GETFIL      GET FILE SPEC
A11A 25 63                  BCS     LIST9       ANY ERRORS?
A11C 86 01                  LDA A   #1          SET UP CODE
A11E A7 00                  STA A   0,X         OPEN FOR READ
A120 86 00                  LDA A   #0          SET UP BINARY
A122 BD AD 33               JSR     SETEX1      SET BIN EXTENSION
A125 BD B4 06               JSR     FMS         CALL FMS
A128 26 55                  BNE     LIST9       CHECK FOR ERRORS
A12A 86 FF                  LDA A   #$FF        SET FF FOR BINARY
A12C A7 3B                  STA A   59,X        SUPPRESSION FLAG
A12E 86 00                  LDA A   #$00
A130 B7 A1 E0               STA A   TEMP5       SET UP BYTE COUNTER

                    * BINARY STORE ROUTINE *

A133 CE A8 40       LIST4   LDX     #FCB        POINT TO FCB
A136 BD B4 06               JSR     FMS         CALL FMS-GET DATA
A139 26 32                  BNE     LIST6       ERRORS?
A13B BD A1 40               JSR     MOVE        STORE DATA
A13E 20 F3                  BRA     LIST4       REPEAT
A140 FF A1 DA       MOVE    STX     XTEMP2      STORE INX
A143 F7 A1 DF               STA B   TEMP4       STORE B
A146 F6 A1 E0               LDA B   TEMP5       LOAD BYTE COUNTER
A149 5C                     INC B               BUMP THE COUNTER
A14A FE A1 08               LDX     XTEMP1      LOAD INX BASE ADDRESS
A14D A7 00                  STA A   0,X         STORE A
A14F C1 01                  CMP B   #$01
A151 27 00                  BEQ     JUMP        PASS INX (X)
A153 C1 02                  CMP B   #$02
A155 27 09                  BEQ     JUMP        PASS INX (X)
A157 C1 03                  CMP B   #$03
A159 27 05                  BEQ     JUMP        PASS INX (X)
A15B C1 04                  CMP B   #$04
A15D 27 01                  BEQ     JUMP        PASS INX (X)
A15F 08                     INX
A160 FF A1 08       JUMP    STX     XTEMP1      STORE INX
A163 F7 A1 E0               STA B   TEMP5       STORE COUNTER
A166 FE A1 DA               LDX     XTEMP2      GET OLD INX
A169 F6 A1 DF               LDA B   TEMP4       GET OLD B
A16C 39                     RTS                 GO BACK FOR MORE

                    * DISK CLOSE ROUTINE *

A160 A6 01          LIST6   LDA A   1,X         GET ERROR STATUS
A16F 81 08                  CMP A   #8          IT IT EOF ERROR?
A171 26 0C                  BNE     LIST9
A173 86 04                  LDA A   #4          CLOSE FILE CODE
A175 A7 00                  STA A   0,X         STORE IN FCB
A177 BD B4 06               JSR     FMS         CALL FMS-CLOSE FILE
A17A 26 03                  BNE     LIST9       ERRORS?
A17C 7E AD 03               JMP     WARMS       RETURN TO FLEX
A17F BD AD 3F       LIST9   JSR     RPTERR      REPORT ERROR
A182 BD B4 03               JSR     FMSCLS      CLOSE ALL FILES
A185 7E AD 03               JMP     WARMS

                    * MESSAGE STRING *

A188 40             BEGIN   FCC     ' OVE BINARY FILE TOT '
A189 4F 56
A18B 45 20
A18D 42 49
A18F 4E 41
A191 52 59
A193 20 46
A195 49 4C
A197 45 20
A199 54 4F
A19B 3F 20
A19D 20                     * IN HEX ROUTINE *

A19F BD 0C          INPUT   BSR     GETHEX
A1A1 B7 A1 DC               STA A   TEMP1
A1A4 BD 07                  BSR     GETHEX
A1A6 B7 A1 00               STA A   TEMP2
A1A9 FE A1 DC               LDX     TEMP1
A1AC 39                     RTS

A1AD BD 10          GETHEX  BS      GETADD
A1AF 48                     ASL A
A1B0 48                     ASL A
A1B1 48                     ASL A
A1B2 48                     ASL A
A1B3 16                     TAB
A1B4 BD 09                  BSR     GETADD
A1B6 18                     ABA
A1B7 16                     TAB
A1B8 FB A1 DE               ADD B   TEMP3
A1BB F7 A1 DE               STA B   TEMP3
A1BE 39                     RTS
A1BF BD AD 15       GETADD  JSR     GETCHR
A1C2 80 30                  SUB A   #$30
A1C4 2B 0F                  BMI     OUTS
A1C6 81 09                  CMP A   #$09
A1C8 2F 0A                  BLE     RERUN
A1CA 81 11                  CMP A   #$11
A1CC 2B 07                  BMI     OUTS
A1CE 81 16                  CMP A   #$16
A1D0 2E 03                  BGT     OUTS
A1D2 80 07                  SUB A   #$07
A1D4 39             RERUN   RTS
A1D5 7E AD D3       OUTS    JMP     WARMS

                    * RAM STORAGE *

A1D8                XTEMP1  RMB     2
A1DA                XTEMP2  RMB     2
A1DC                TEMP1   RMB     1
A1DD                TEMP2   RMB     1
A1DE                TEMP3   RMB     1
A1DF                TEMP4   RMB     1
A1E0                TEMP5   RMB     1
                    END

NO ERROR(S) DETECTED
```


Georgia Institute of Technology
Atlanta, Georgia 30332

Doo Williams
68 Micro Journal
5900 Cassandra Smith Road
Hixon, Tennessee 37343

Dear Don:

We are working on some interface and furnace controller experiments for one of our laboratories. Currently, we use a SWTP System and have our experiment modules on the 6-30 bus. We would like to make several 6801 based stand-alone modules so there will be more access to these experiments. We need a tiny BASIC (2K) with peek and poke to simplify the student - computer interface (we don't have time to teach assembly language or machine language in this course). Tom Pitman's early version would be satisfactory, if in EPROM. Do you know of anyone who is using such? We cannot really justify more than one 2K or one 4K (EPROM) for the BASIC interpreter, and do not need floating point.

We have a simple PID control algorithm and simulated furnace written in TRS Basic we use to demonstrate "tuning" a controller. We will be pleased to share our experiments with anyone.

Cordially,

Joseph L. Pentecost, Director
School of Ceramic Engineering

JLP:dpp

Enclosure

William Hartmann
RR 2 Box 121-1
Blue Springs, MO 64015

April 12, 1982

Dear Sir:

When I first converted my system to a 6809 FLEX system I was working with a single 5 inch drive. I used a single drive, multiple file copy program, "MCOPY" from

Conejo Computer Products. This program was very well written and uses FLEX to its maximum to speed up the copy operation. When copying a file MCOPY keeps the original file creation date. I find this convenient to tell which files were current and which had been updated.

Enclosed is a program "DATECOPY" which patches the FLEX COPY.CMD so that the orginal file creation date is retained during the copy. It is written for the 6809 version, but could be used a 6800 if the address shown in the listing can be found.

To create the modified COPY assemble DATECOPY then append

```
+++ APPEND,COPY.CMD,DATECOPY.BIN,COPY.CMD
```

```
2                       ********************************************************
3                       *
4                       * THIS IS A OVERLAY TO THE FLEX09 COPY.CMD VER 1
5                       * TO MAINTAIN THE ORIGINAL FILE CREATION DATE
6                       * DURING COPY
7                       ********************************************************
8
9
10                      *
11                      *  ORIGINAL CODE
12                      *
13                      * C1A1 7E CD03   JMP  WARMST ($CD03) RETURN TO FLEX
14
15                      * C351 B6 01     LDA $$01
16                      * C353 A7 B4     STA ,X
17                      * C355 BD D406   JSR  FMSCAL ($D406) OPEN FOR READ
18                      *
19
20  C102                        ORG    $C102
21  C102 02                     FCB    2          MAKE VERSION 02
22
23  C1A1                        ORG    $C1A1      PATCH CLEANUP
24  C1A1 7E  C532               JMP    CLNUP
25
26  C355                        ORG    $C355      PATCH FILE OPEN
27  C355 BD  C541               JSR    DSAVE      TO SAVE DATE
28
29
30                      * FLEX EQUATES
31              CD03    WARMST  EQU    $CD03
32              D406    FMSCAL  EQU    $D406
33              CC0E    FDATE   EQU    $CC0E      FLEX DATE BUFFER
34
35
36  C520                        ORG    $C520      AREA PAST COPY
37
38  C520                DBUF    RMB    3          DATE SAVE BUFFER
39  C523 FC  CC0E       START   LDD    FDATE      SAVE CURRENT DATE
40  C526 FD  C520               STD    DBUF
41  C529 B6  CC10               LDA    FDATE+2
42  C52C B7  C522               STA    DBUF+2
43  C52F 7E  C100               JMP    $C100      GO TO COPY
44
45                      * RESTORE DATE ON EXITING
46  C532 FC  C520       CLNUP   LDD    DBUF
47  C535 FD  CC0E               STD    FDATE      STORE IN FLEX DATE BUFFER
48  C538 B6  C522               LDA    DBUF+2
49  C53B B7  CC10               STA    FDATE+2
50  C53E 7E  CD03               JMP    WARMST     RETURN TO FLEX
51
52                      * GET FILE CREATION DATE
53  C541 BD  D406       DSAVE   JSR    FMSCAL     OPEN FILE
54  C544 34  07                 PSHS   CC,D       SAVE ERROR CONDICTION
55  C546 EC  88 19              LDD    25,X       GET FILE CREATION DATE
56  C549 FD  CC0E               STD    FDATE      AND SET IN FLEX
57  C54C A6  88 1B              LDA    27,X
58  C54F B7  CC10               STA    FDATE+2
59  C552 35  87                 PULS   CC,D,PC    RESTORE ERROR FLAG AND EXIT
60
61                          END    START
```


HELP

Dear Sirs:

I bought PIMS from Scelbi Publications and have it all working except for the load from cassette command. I would appreciate if someone could help me with it, I have a SWTPC 6800/2 with version 2.3 basic.

Thank you,
Richard Price
73090 McKay, Romeo, MI 48087

***

SWTPc 8K BASIC 2.3

Need HELP in locating someone who can furnish source or reference points for SWTPC BASIC Ver. 2.3. I have tape only; it prompts, but will not accept any statement. RUN usually linefliks a couple times and then to "READY" prompt. If no source, I would appreciate a 'dump' of the code so I can check against my copy. Thanks for any help.

Ray Baumiller
1696 4th St. East, Monongahela, PA 15063

***

CLASSIFIED

6800 CPU Card $100.00. AC-30 Cassette Interface w/documentation $50.00. DC-3 Controller Card $100.00.
LEW 1-615-842-4601

***

Tano 56K Dual Serial Ports, Duel 5" Disk Drives, complete software package. Like New, Wholesale $5300 will take $2995. Three SWTPC 8K memory boards $95.00 ea. PR-40 Printer, Extra Paper, And Ribbons, $95.
LEW 10am-5pm 1-615-842-4600

***

For Sale: Tano 48K 6800 With Dual Serial Ports, Dual 5" Drives, Electric Word Processor, TSC Editor and XBasic, FLEX Dos $2395.00
Charles Skeleton (805) 484-3162.

***

SWTP MP-8M Memory Board (8 K) $50.00; GIMIX 16K Memory Board-each 4 K block switch-addressable to any 4 K boundary $100.00; two unused HM6116P-3 static (2Kx8) RAM chips ea. $10.00.
Contact Ches or Andy at 301-422-7753 5:00 to 10:00 pm.

***

SWTPC-24K, Percom Disk System, CT-64, Video Monitor, AC-30, MP-S, MP-C, Software.
M. Kochyan, 20901 Crestmont, Dearborn Heights, MI 48127 (313) 271-3594

***

SWTPC AC-30, 3-MP-M, MP-B, MP-A2, MP-T, MP-R, 2-MP-LA, MP-S, Lucidata Pascal V3, FLEX 2 w/utilities & Basic, Miniflex w/utilities& Basic, MD-1 Disk Driver, SA-400 Disk, 48K DM-64, PMB-1 SWTBUG, Fadbug $1200 or make offer on system or single boards.
Rick Richardson 6316 Old Melrose Rd, Roseburg, OR 97470

***

SWTP MP-68, 32K, MP-S, MF-68, AC-30, Lots of Disk Software, all documentation. $1000.
Marty Frankel, 203-322-4533 Eves.
52 Calass Ln, Stamford, CT 06903

***

BUY ONE OF OUR BOARDS AND STUFF IT!

$19.50 EACH
(Offer Expires June 30th)

- Dual Serial I/O Board (30 Pin)
- 30 Pin Extender Board
- 50 Pin Extender Board

(Both extender boards fully labeled and shielded to prevent RF interference)

$29.50 EACH
(Offer Expires June 30th)

- 6809 CPU Board
- Multiple I/O Board
- Video Display Board
- Video Color Graphics Board
- 5¼" Disc Controller Board (50 Pin)

$39.50
(Offer Expires June 30th)

- 64K Dynamic Ram Board
- Comletely Sold
- A
- Fu

See Write up, this Issue, Page 20

$59.50
(Offer Expires June 30th)

- Mother Board
- .093 Thick
- No "Flex" Problems
- Full Documentation Included

- All Boards Solder Masked Both Sides
- All Silk Screened Nominclature
- Full Documentation Included

- All Boards Available Fully Stuffed
- VISA & Master Card Accepted
- Add $3.50 for shipping & handling

You're Right

We can't sell boards at this price for very long. But what better way to introduce you to DATA SYSTEMS "68". Order before June 30, 1982 (call 305-425-6800 collect for credit card orders) and realize savings of up to 50%. We guarantee your satisfaction.

DS 68

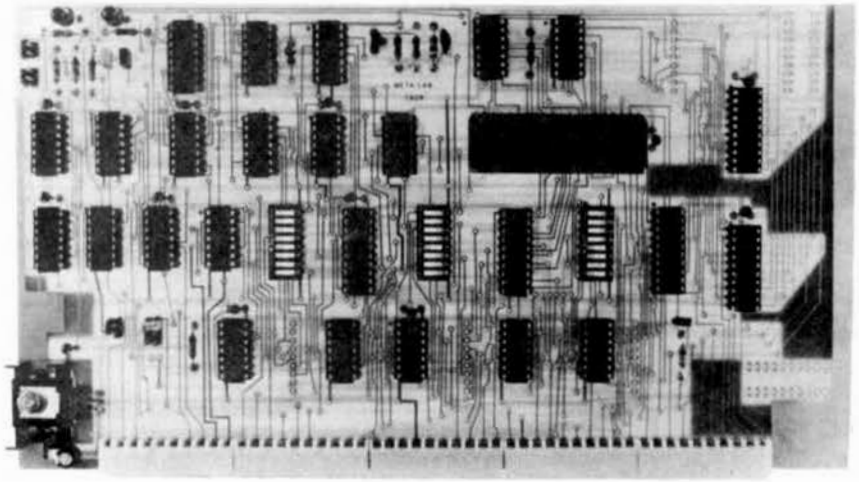
Data Systems "68"
2316 Diversified Way
Orlando, Florida 32804
(305) 425-6800

Limited Warranty

All DS68 products are guaranteed for a period of 90 days to be free of defects in materials and workmanship or return to Data Systems "68" for free replacement.

TERMS: Check, MasterCard, Visa. Sorry, NO C.O.D.'s. Florida residents add 5% sales tax. Specify type board(s) and quantity of each when ordering.

CP/M ON THE SS50 BUS



Why CP/M?

We all know that productivity is increased with good software. Most of us have found that we have more fun when using software correctly designed for our specific tasks. CP/M has over 2000 application programs. **With CP/M you can get the exact program to fit your needs and budget.** There is also a large selection of programs available for the cost of the media through the CP/M users' groups. **The latest developments are available under CP/M—even an ADA language subset.** Simply put, the application programs available under CP/M are unequalled.

How to install the Z809

Plug the Z809 Board in an unused SS50 slot. **On most SS50 systems, just boot the system using the CP/M disk and you are up and running.**

How does the Z809 Softboard System work?

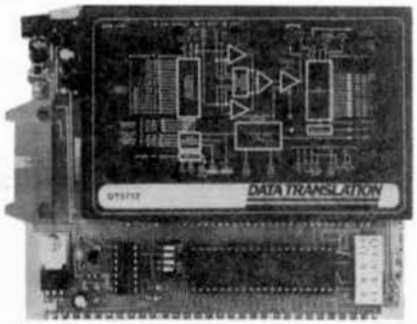
All I/O functions of the Z809 are directed through the 68XX system CPU board. **Use your current disk controller board, CPU, memory, disk drives, printer, and terminal with standard SS50 configurations.** The Z809 does not affect use of the system CPU. The Z809 does not replace your CPU board. It uses your 68XX as an I/O processor. To change back to your current operating system, just boot the system with the appropriate disk.

Other information on the Z809

META LAB has designed the basic input output system (BIOS) of CP/M to take full advantage of the physical characteristics of your disk drive. The BIOS drivers allow you to dynamically specify all characteristics of your diskettes and drives. The software is reconfigurable when you expand your system with more memory or hard disks. **The Z809 is compatible with standard CP/M formatted disks so that software is easily exchanged or ported to other systems.** The board runs at 4 Mhz when run on a 2 Mhz SS50 system. It executes Z80/8080 object code. Mminum memory requirements are 24K, however some application programs require 56K. The Z809 supports up to 56K of system memory. The Softboard is designed to be an expandable and adaptable part of your total computer system.

Breakthrough in SS50 Computing

The Z809 is the most useful peripheral board you can add to your system. We've put the best of both worlds together for you. Each Z809 Softboard System comes with a 50 pin processor board, user manual, CP/M 2.2 operating system specifically tailored to your CPU and disk controller board, and complete CP/M reference manuals. Total System Cost $595



Products Coming from Meta Lab

Dealer/OEM Inquiries Invited

Copy Utilities — Copy any type of file between CP/M, FLEX, OS9
Copy any type of file between OS9 and FLEX

6805 Development System IEEE488 — Complete Talker, Listener, Controller

DATA ACQUISITION AND PROCESS CONTROL

ADC 1200	High Speed 12 Bit A/D
DAC 1220	12 Bit D/A Board—2 Channels
ADA 800	Eight Channel 8 Bit A/D and D/A
DAC 820	8 Bit D/A Board—2 Channels

ORDERING INFORMATION

Shipping & Handling $5
Terms Prepaid
COD Certified Check
VISA and MC

CP/M is a registered trademark of Digital Research Corp. FLEX is a registered trademark of Technical Systems Consultants Inc. OS9 is a registered trademark of Microware.

META LAB

Suite 106 2888 Bluff St.
PO Box 1559 Boulder, CO 80306
303-499-4236

ALFORD'S GOOD NEWS!

UNLIMITED COMPUTER SPEECH...

Our speech synthesizers allow your computer to talk, LITERALLY! Using our VS-1 (for the SS-50 bus) or the SP-1 (for the COLOR COMPUTER), your programs can now communicate with speech! You can add a new dimension to almost any application with our synthesizers. Business programs, computer assisted instruction, industrial control, alarm systems, most anything can benefit from the power of speech.

We don't just sell you a hunk of hardware, either. Our synthesizers come with well-documented manuals detailing speech theory, programming examples, sample dictionaries, coding tables and more. We even include software to help to integrate these products into your system! Our unique "VOX-EDITOR" speech editing program, for instance, allows you to create and edit speech messages simply and easily. You can save the speeches created into files which can be directly read by Basic or other programs.

Both units come completely assembled and tested. No electronics knowledge or assembly is required. The SP-1 plugs directly into the ROM-PACK slot on the COLOR COMPUTER. Audio is fed back into the computer and routed to your television monitor. The VS-1 occupies one slot on the 30-pin I/O bus, and only requires the attachment of an external speaker (not supplied) for operation.

The VS-1 software is available for SSB DOS68 and all versions of 6800 FLEX (except mini-FLEX), and for SSB DOS69, FLEX-09 and OS-9, and may be ordered on 5- or 8-inch disk. The SP-1 software comes on a single cassette (disk versions not available at this time).

TEXT TO SPEECH TRANSLATOR...

We are now offering a speech translator for the VS-1 and SP-1 synthesizers.

The SC-1/VS-1 translator allows you to generate speech files from the keyboard, or from standard disk text files. The resulting file may be "fine-tuned" with the VOX-EDITOR, thus greatly reducing the amount of time needed to generate custom messages. Also, the SC-1/VS-1 translator allows you to simply speak your text files, allowing you to listen to your work while you do something else, eliminating the need to keep your eyes "glued" to the screen. You may also type from the keyboard directly to the translator.

The SC-1/SP-1 translator allows you to operate directly from Color Basic by adding a new port to Basic. Now, by typing "PRINT #1, "THIS IS A TEST", your computer will say "THIS IS A TEST"! By using control codes (PRINT #1,CHR$(xx)), you can make the translator pronounce everything which goes to the screen!

Flags are included in both versions which will cause the translator to pronounce punctuations. In this way, you can listen to every "jot and tittle" as it is read!

We plan to offer regular updates at a reduced price, so that as we continue to improve the translational accuracy of this product, you can take advantage of the updates without having to pay the full price!

The SC-1/VS-1 version is available for 6800 SSB DOS68, versions 5.1 and 6.0, and for all versions of FLEX (except mini-FLEX). 6809 versions are available for DOS69, FLEX-09 and OS-9. All may be ordered on 5-inch or 8-inch disk. The Color Computer version requires 16k or 32k memory (Extended Color Basic not required), and is supplied on cassette only.

The disk versions of the SC-1 translator include the source files, so you can make custom modifications to allow you to use it with other speech boards (we admit they exist!), or to change I/O addresses, etc.,. The 6809 disk version is self-relocating to place it at the top of available memory and out of the way of your other programs!

PUBLIC NOTICE...

FLEX is a trademark of Technical Systems Consultants, Inc. OS-9 is a trademark of Microware Systems Corporation. Radio Shack and TRS-80 Color Computer are trademarks of the Tandy Corporation. SSB ain't so picky! (love you, Rick, Larry, Jim, Debbie, Ken, etc...!). SCREDITOR III (no, Ed, we aren't gonna change it!) is a trademark of Alford & Associates, but then who else would use the name? For that matter, who can pronounce it?

TREK...

Some of you might not know it, but many of the games which you have seen written in Basic are emulations of games which were developed with compilers or assemblers for large machines. Some are pretty engrossing games, but all share two very bad problems ... question-and-answer play, and low speed. We went in the other direction. We took one of the most popular Basic games of all time, re-wrote it completely in assembly language, using full interrupt-driven, multi-tasking structure, and came up with what we feel is one of the best games around today!

Although our TREK uses alphanumerics exclusively in its display (thus making it compatible with most every memory-mapped display around), the lack of hi-res graphics is not missed due to the edge-of-the-seat action! The enemy space destroyers use multiple battle plans as they pursue you around the quadrants. Torpedoes destroy anything in their paths, forcing you to take continual evasive action. Damage control reports flash on and off as you take hits and repairs are made. Energy runs down continuously, making it impossible to sit back and relax. The only way to believe that so much action could be fitted on one screen is to play it!

TREK-68/6800 is available for 6800 processors running SSB DOS68 or any version of FLEX (except mini-Flex), and may be obtained on 5- or 8-inch disk. TREK-68/6809 is available for 6809 processors running SSB DOS69 or FLEX-09, and can be shipped on 5- or 8-inch disk.

COLOR-TREK is available on cassette for the TRS-80 COLOR COMPUTER, and requires 16K or more memory (Extended Color Basic NOT required). COLOR-TREK+ is identical to COLOR-TREK, except that it requires our SP-1 Speaker Pack voice synthesizer for operation, so the computer uses english speech messages continuously throughout program play.

TREK-68 requires a MEMORY-MAPPED display for operation as well as a source of interrupts (such as the MP-T timer) programmed for 50-120 interrupts per second. Also, a serial or parallel keyboard port is required (will not run with the MP-C interface).

FANTASTIC PROOFREADER...

We are now offering Peter Stark's SPELL'N FIX proofreading program for both FLEX and SSB! We think that this program is the best spelling checker/corrector available today, and is definitely the fastest and most flexible such program that we have used.

SPELL'N FIX provides a number of features that we have not seen on any other such program ... features such as multiple dictionaries, compressed dictionaries (25000 words on a single 5-inch diskette!), automatic dictionary updates, the option of simply printing errors, marking them in the file being proofread, or correcting them! Sector limits may be set to limit the length of file on each disk, text files may be split up, the dictionary split, and best of all, SPELL'N FIX doesn't let you down with DISK FULL errors! When the disk fills up, SPELL'N FIX automatically closes the output file and asks for another disk on which to continue!

In addition, you can perform most FLEX commands directly from SPELL'N FIX. The SSB version includes a directory list command which allows you to list disk files directly from SPELL'N FIX. This means that you don't have to stop and restart simply because you have forgotten a file name in the course of proofreading!

SPELL'N FIX is available for SSB DOS68 versions 5.1 and 6.0, and DOS69 (all versions), FLEX-9, 6800 FLEX (all versions except mini-FLEX), and can be ordered on 5- or 8-inch disk. Versions for mini-FLEX and PERCOM available directly from STAR-KITS. OS-9 versions will be available soon.

ALFORD & ASSOCIATES
P.O. BOX 6683
RICHMOND, VA., 23230
804·320·6722

SSB UTILITIES...

We are now shipping our UTILITIES #1 disk for DOS68 and DOS69 users. If your disk library is getting out of hand, or if you are going bonkers trying to make sense out of fifty, one hundred, or more files on a disk, this set of programs is for you!

We have developed a method to write a unique I.D. sector on SSB disks. In this sector, we include an 8-character disk access code, a 16-character disk name, an 8-character disk I.D., an 8-character disk creation date, an 8-character disk update date, and a 64-character comment field. Our NAME utility prompts you for the information and builds the information sector.

UPDATE then allows you to change the I.D. sector selectively. You can, without the access code, change only the update date and comments field, allowing you to make a quick change to reflect, say, the most recent change to the files on the disk. Using the ACCESS code, you can change everything except the disk creation date.

LOCK allows you to list- delete- and write-protect your files. With list protection, the list-locked files only list when you supply the disk access code!

LIST is a multi-option program, allowing you to list transients, non-transients, locked files (if you supply the disk access code), free space, disk information, or any combination of these.

PURGE is a quick and easy mass file deletion program. Each file on the disk is presented for your inspection. Multiple prompts then determine whether or not to delete each file.

TITLE generates BIG LETTER banner pages for your printed listings. You supply a name for the banner page of up to twelve characters, and TITLE generates the banner page with the current date and time (as contained in YDATE and YTIME).

DUMP is a formatted hexidecimal and ASCII memory dump. When directed to a hard copy device, page boundaries are honored to make dumps more readable.

COUNT allows you to quickly determine file sizes in lines and characters. Multiple files may be specified to allow you to count in a single pass those jobs which you have spread over several files.

UTILITIES #1 is available for DOS68, versions 5.1 and 6.0, and for DOS69 (all versions), and may be ordered on 5- or 8-inch disk. The manual also includes some very good information on the chained sector technique which SSB uses on their disks.

PRICES...

TREK-68/6800 disk, manual	$ 29.95
TREK-68/6809 disk, manual	$ 29.95
TREK-68 manual only	$ 12.95
COLOR-TREK cassette, manual	$ 17.95
COLOR-TREK+ cassette, manual	$ 24.95
COLOR-TREK manual only	$ 9.95
SPELL'N FIX disk, manual	$ 89.29
UTILITIES #1 disk, manual	$ 29.95
UTILITIES #1 manual only	$ 12.95
VS-1 complete, disk, manual	$194.95
VS-1 complete, disk, manual (less SC-01A)	$134.95
VS-1 disk, manual (use VOX-EDIT with brand X)	$ 39.95
VS-1 manual only	$ 19.95
SC-01A synthesizer chip	$ 65.00
SP-1 pack, cassette, manual	$179.95
SP-1 manual only	$ 14.95
SC-1/6800 disk, manual	$ 39.95
SC-1/6809 disk, manual	$ 39.95
SC-1/COLOR cassette, manual	$ 29.95
SCREDITOR III disk, manual, tutorial cassette	$100.00
SCREDITOR III manual and cassette only	$ 29.95

THE BEST NEWS...

God is truly a God of miracles. We could not begin to enumerate the miracles which He has performed for us, and wish to give thanks for His love and care in our lives. Without Him, we would not be in business. We hope that our concern and devotion to our customers reflects our love for Him. We do make mistakes, and occasionally an order is loused up or lost, but we'll do everything we can to correct any mistakes that we make. Thank you for your patronage.

SCREDITOR III DOES IT ALL...

This ad is being prepared using the SCREDITOR III word processor. Unlike any other editing package available for 6800/6809 machines, SCREDITOR III supports true multiple column editing capability. This ad was typed a column at a time. No cut and paste was used to prepare it (well, one, actually... our printer only prints 132 columns wide!).

SCREDITOR III is a true screen editor. What you are reading now is what we saw as we were entering the text! With full cursor movement capability, automatic justification, true multi-column insert, delete, paragraph splitting and melding, and edit capability of up to 251 character positions per line, SCREDITOR III makes document preparation a breeze! Also, you can define page footers and headers, automatically, delete and then re-insert them, or change them at any time.

You can merge multiple files with SCREDITOR III. You can even specify where to insert a file, what line in the file to start from, and how many lines of the file to use! You can extract or copy lines from the current file into another file. You can even do form letter printing with automatic line item merging for custom mailings.

Automatic justification (left, center, right, or full line) can be turned on or off at any time. Margins can be moved at any time. Just about any job can be done better with SCREDITOR III.

You can define "symbols" with SCREDITOR III. Up to 251 keystrokes may be included in a single symbol, including all edit and line commands, and later recalled with a single keystroke. Up to 26 symbols may be defined, and the symbols may be saved to a disk file for later recall.

Unlimited tabs may be set with SCREDITOR III, and may be changed at any time as you desire. Margins and tabs are continuously displayed for quick reference, along with all major edit modes, current column, current line, total lines, current display columns, etc...

SCREDITOR III runs with practically any SERIAL TERMINAL or with most any MEMORY MAPPED DISPLAY. If you change from a serial terminal to a memory-mapped display, or if you simply change display or terminal types, you don't have to buy a new copy of SCREDITOR III. Just run SYSGEN and define your new hardware! Even key definitions may be changed by SYSGEN! Terminals with special function keys may also be accommodated. Simply run SYSGEN, and hit each key as it asks you to, and SCREDITOR III will use that key in your system!

SCREDITOR III is designed to provide capabilities for any office, personal, programming, or other editing and formatting job that you might have. The applications are essentially unlimited.

The manual provides extremely detailed information about every mode, feature and function of SCREDITOR III, and was written with office personnel and the unsophisticated user in mind. A unique feature of SCREDITOR III is a tutorial cassette which makes learning to use SCREDITOR III simpler than any other such program on the market. The cassette takes you, step-by-step, through every function of SCREDITOR III, giving you practical examples and learning sessions with test files. No more reading for three weeks to learn how to do it!

SCREDITOR III is available to run under DOS68 versions 5.1 and 6.0 and all versions of 6800 FLEX (except mini-FLEX), and for DOS69, FLEX-9 and OS-9. SCREDITOR III may be ordered on 5- or 8- inch disk.

GENERAL INFORMATION...

Unless specified, all programs (except Color Computer) shipped on 5-inch disk for FLEX-9. Virginia orders add 4% sales tax. We ship by UPS on all domestic orders unless otherwise specified. COD's MUST go by UPS. Overseas orders shipped by air parcel post. On North American orders under $100, add $5 for shipping and handling. We pay shipping on orders over $100. On overseas orders, regardless of order value, add $10 for shipping. ORDERS RECEIVED WITHOUT SHIPPING WILL BE RETURNED UNFILLED.

We accept MASTERCHARGE, VISA and personal checks. Personal checks require two to three weeks to clear before processing can proceed. All checks must be in U.S. funds drawn on U.S. banks. Dealer and OEM pricing and licensing agreements are available. Please CALL for details. Open accounts to Fortune 1000's, schools, established accounts only. We regret this policy, but can no longer finance other businesses.

ELEKTRA

COMPUTER PRODUCTS

ELEKTRA CABINET

Made of heavyweight 0.090" thick aluminum.
Interior Size: 18-1/2" wide by 21-7/8" deep x 6-3/4" high
Heavy duty A.C. line cord.
A.C. fuse holder
EMI filter
Fan with filter
Back panel has 10 cutouts for "D" type data connectors
Front panel has key on/off power switch, 2 illuminated push buttons (Reset and NMI/Abort), and 2 cutouts for 5-1/4" floppy drives.

Without power supply:	$260.00
With 110v power supply, disk regulator board or two filler plates:	$460.00
With 220v power supply, disk regulator board or two filler plates:	$510.00

POWER SUPPLY

Highest quality linear power supply conservatively rated at 15a @ 5v, 3a @ 16v, 3a @ -16v
3 primary inputs to adjust for light, medium, and heavy loading

110v version: $175.00 220v version: $225.00

DISK REGULATOR BOARD with cables

Powers 2 5-1/4" floppy drives: $50

Filter (fan): $10.00

ELEKTRA CPU 8/9

Choice of 6802, 6808 or 6809 CPU
(6802 and 6808 are software compatible with the 6800 at the opcode level).

DEVICE		6809 ADDRESS	6802-6808 ADDRESS
3 2716 Eproms	Eprom #3	F800-FFFF	F800-FFFF and E000-E7FF
	Eprom #2	F000-F7FF	F000-F7FF
	Eprom #1	E800-EFFF	E800-EFFF
1K Scratchpad RAM		E400-E7FF	A400-A7FF and A000-A3FF
MC6840 Triple Timer		E210-E217	8210-8217

MC14411 Baud Rate Generator producing baud rates of:
Low Range 110, 150, 300, 600, 1200, 4800, and 9600
High Range 440, 600, 1200, 2400, 4800, 19200, and 38400
The board does not contain a DAT and does not support extended addressing.
The board supports DMA by either HALT or BUSREQ when a 6809 CPU is used.
DMA to the devices on the CPU card is not supported.

The board will run any of the MIKBUG™ compatible monitors in the 6802-6808 mode and SBUG-E, NUMBUG, and GIMIX-09 in the 6809 mode. The ELECTRA CPU 8/9 will run any of the popular disk controller boards with the appropriate software. Special versions of OS-9™ L1 are available.

Bare board: $50.00* Kit: $225.00* Assembled: $275.00

ELECTRA DPS Dual Port Serial Card

Fits the standard 30 pin SS-50 bus I/O slot
Can be configured for 4 addresses per port with the B port 2 addresses higher than the A port or for 16 addresses per port with the B port 4 addresses higher than the A port.
Each port is terminated at two 16 pin dip sockets, one socket configured for modem and the other socket configured for terminal or printer. RTS, CTS, DTR, DCD, DTR are appropriately implemented.
Each port has independent selection of baud rate.
Each port allows the interrupt request to be jumpered to the IRQ or FIRQ/NMI bus line.

Bare board: $20.00* Kit: $60.00* Assembled: $80.00

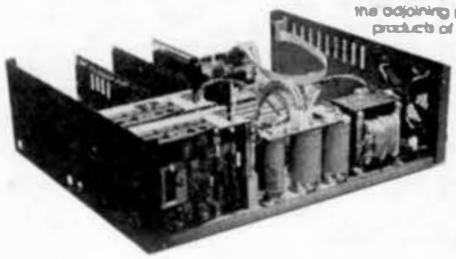
Assembled cable (two required for each interface board) $20.00 each

ELEKTRA DPP Dual Port Parallel Card

Fits the standard 30 pin SS-50 bus I/O slot.
Can be configured for 4 addresses per port or 16 addresses per port (occupying the first four addresses of the I/O slot).
The direction of the TTL buffers can be controlled by either on board jumper connection or by a signal from the peripherals.
The interrupt request lines for each port may be individually jumpered to the IRQ or FIRQ/NMI bus line.

Bare board: $20.00* Kit: $60.00* Assembled: $80.00

Assembled cable (two required for each interface board) $20.00 each



The CPU, 56K memory board, and DMA controller board in the adjoining picture are products of GIMIX, Inc.

ELEKTRA Motherboard

Heavyweight 0.125" thick, 18" long by 9" wide.
11 memory (50 pin) slots, 4 or 8 slots may be cut off for shortening to 14" and 10" lengths respectively.
8 I/O (30) pin slots.
1" spacing between all memory and I/O slots.
On board baud rate generator with high and low ranges providing jumper selectable rates for each of the five I/O baud rate lines.
Complete address decoding and selectability for the I/O ports.
Choice of 4, 8, or 16 addresses per I/O port.
Slow device circuitry permitting 1 MHz 30 pin disk controllers to run with 2 MHz 50 pin CPU boards.
Extended addressing capability for meeting SS-50C bus specifications.

Bare board: $80.00 Kit: $240.00** Assembled: $300.00**

**Gold square pin connectors instead of tin add $80.00

ELEKTRA Chasis

Includes cabinet, power supply, disk regulator board and power cables, and assembled motherboard with gold connectors, totally installed. Ready to use with documentation.

$825
Add $50.00 for 220V

*WARNING

AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have manufactured our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controller(s), drives, and software along with I/O terminal(s) to form working computer system(s).

AAA Chicago Computer Center

120 Chestnut Lane • Wheeling, IL 60090 • (312) 459-0450

Closed June 26 thru July 18

Phone consultation available most weekdays from 4 PM to 6 PM

SMOOTH™ Software

SUPER MODEM PROGRAM

Transmit manually to distant computer

Transmit disk files (text) of any length to distant computer

Receive and save disk files (text) of any length on local disk system (Sending computer does not support X-on/X-off protocol, then the received files are limited in size by the computer memory.

Tested to transmit and receive text at speeds up to 9600 baud. (CRT terminal must be capable of operating at a baud rate higher than the one the modem is operated at.)

Half duplex option in case distant computer doesn't echo

Echo option so user can simulate a time sharing system. (Super Modem Program doesn't support auto-answer but the source is provided for those individuals who wish to adapt our program to their special needs.)

Replaces CR with CR/LF (user option) for those using time sharing systems that don't transmit LF's.

Slow disk file transmit (user option) based on character verification for use on time sharing systems to which disk files cannot be sent at speed suggested by the baud rate.

Please specify 6800 SSB, 6800 FLEX™, or 6809 FLEX™, 5" or 8"
Manual and disk with both source and object code $75.00

STANDARD MODEM PROGRAM

Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Receive textonal disk files is limited to those small enough to completely fit within the receiving buffer.

Please specify 6800 SSB, 6800 FLEX™, or 6809 FLEX™, 5" or 8"
Manual with instructions, source listing, and flow chart; disk with both source and object code $45.00

ALL IN ONE
Editor - Text Processor - Mailing Labels
Mailing Lists - Use any CRT terminal and printer

Supports Editing commands such as bottom, change, delete, find, insert (single line), input (multiple lines), list, next, overlay (with cursor editing), character deletion and insertion), overstrike (for selected darker text), print, restart, set, top, underline, up, and verify.

Supports Text Processing commands such as block copy, block move, centering, margin justification (widen and narrow), paging, and tabbing.

Mailing Lists and Labels. Use the same mailing list disk file (with protected areas) for both mailing labels and repeat letters. Repeat letters are personally adressed to each person or selected persons on the mailing list.

Most Powerful File Handler found in any editor. Append one file to the end of another, or insert (merge) one file into another as designated by the line pointer. Print specified lines to your printer or to a disk file. Edit files larger than the text buffer. Does not produce output files when not desired. Delete disk files from the editor.

Printer commands. Control characters can be sent to the printer for formal control either directly from the control terminal or by imbedding them in the text. The set command contains interface initialization and character output routines to support the SWTPC MP-C interface as well as the standard serial and parallel interfaces. JumPs are also provided to user supplied Printer routines. User selects the port address (0 thru 7, A or B) thereby eliminating the need for the user to install printer software routines. Editor can be initialized for either 4 or 16 addresses per port.

Editor allows exiting to either the monitor or DOS and then reenter (Warm Start) without destroying previously prepared text in the buffer. The Restart command erases contents of the buffer without the user having to reload the Editor.

The Editor allows the user to toggle between full duplex (no echo) and half duplex (echo) as needed. It responds to commands in both upper and lower case and can be used to create assembler source code and Basic programs as well as text.

Specify 6800 SSB, 6800 FLEX™ 6809 FLEX™, 5" or 8" 50.00
Printed source listing is available for an additional 35.00

Software by Technical Systems Consultants, Inc.

Flex™ (includes Editor and Assembler)	150.00
UniFLEX™ (includes one year maintenance and update)	450.00
Editor	50.00
Assembler	50.00
68000 Cross Assembler on 6809	250.00
6809 Cross Assembler on 6800	100.00
Text Processor	75.00
Extended Basic	100.00
Basic Precompiler (specify standard or extended)	50.00
Pascal (Flex™)	200.00
Pascal (UniFLEX™) (Add $75.00 for one year's maintenance and update)	225.00
Sort-Merge Package	75.00
6809 Flex™ Utilities	75.00
6800 Flex™ Utilities	100.00
Debug Package	75.00
Diagnostic Package	75.00

Software by Microware Systems Corp.	UPDATE	SOURCE	MANUAL	OBJECT
OS-9™ Level One Operating System	75.00	400.00	40.00	200.00
OS-9™ Level Two Operating System	75.00	N/A	40.00	500.00
BASIC09™	75.00	N/A	25.00	200.00
OS-9™ Macro Text Editor		300.00	15.00	125.00
OS-9™ Interactive Assembler		300.00	10.00	125.00
OS-9™ Interactive Debugger (Disk version)		100.00	10.00	50.00
CIS Cobol Compiler	250.00	N/A	80.00	900.00
Pascal Compiler	100.00	N/A	40.00	400.00

SWTPC	Kit	Assembled
DMF2C Disk Controller Board (NEW)		595.00
DMF2C Disk Controller Board (Used, very limited quantity)		525.00
SWTPC FLEX™ Disk and manual		35.00
SWTPC FLEX™ Disk without manual		10.00
DC-2 Disk Controller (Limited Quantity)	N/A	125.00
4K RAM Board (Limited Quantity)	N/A	40.00
SBUG-E (2716 compatible)		19.95
SWTBUG (6830 compatible, limited Quantity)		19.95
SWTBUG (2716 compatible)		N/A
MP-S2 Serial interface (dual port)	N/A	120.00
MP-LA Parallel interface (dual port, limited quantity)	40.00	60.00
MP-L2 Parallel interface (dual port)	N/A	120.00
MP-R Single voltage 2716 prom programmer	N/A	114.50
MP-N Calculator board	54.95	92.00
MP-T Interrupt timer	N/A	92.00
MP-8M 8K 6044 Memory board (limited quantity of kits)	150.00	275.00
S32 Universal Static Memory Board	N/A	124.50
MP-09 6809 CPU board	N/A	295.00
69 Chassis, PS, 6809 CPU, 8K, RAM, One Serial Port	N/A	799.00

Universal 68XX Bare Motherboard. 6800/6809, 4/16 addresses per port, 8 50 pin/8 30 pin slots, baud rate generator, 15 1/8" x 9 3/8" 65.00
F &D (bare) Motherboard I, BMB-1 Discontinued
Connectors (10 pin, Titanium-Tin plated 5 microns for near gold quality)
Male with square cross section pins ..... each .50
Female ..... each .75
Gold, Male w/square pins or female ..... each 1.50

SMOOTH™ and **ELEKTRA™** are trademarks of AAA Chicago Computer Center
FLEX and **UniFLEX** are trademarks of Technical Systems Consultants, Inc.
OS-9 and **BASIC09** are trademarks of Microware Systems Corp.
GIMEX® and **GHOST®** are registered trademarks of GIMIX Inc.

ELEKTRA™ SS50 Computer Products

ELEKT - A Dual drive cabinet for 5-1/4" drives with power supply, line cord, fuse, power switch, and power cable to drives	125.00
ELEKTRA Dual drive cabinet, power supply, ps cable for 8" drives	350.00
Cabinet for dual 8" drives only	250.00
Power supply for dual 8" drives only	120.00
PS cables only (Specify brand and type of 8" drives)	30.00

Special Software

4K 6809 HUMBUG	75.00
4K 6800 HUMBUG (RAM needed at $A000 and $D000)	65.00
2K 6800 HUMBUG (With cassette LOAD and PUNCH)	40.00
2K 6800 HUMBUG (Extra commands instead of cassette software)	40.00
Other HUMBUG versions including video versions are available	
Spell'n Fix by Peter Stark	89.29
Dynamite Disassembler	60.00
SUPER SLEUTH Disassembler System	99.00

DISK DRIVES 30 day guarantee, single and double density capability

5-1/4" single head, 40 tracks	$250.00
5-1/4" single head, 40 tracks, flippy	260.00
5-1/4" double head, 40 x 2 tracks	335.00
5-1/4" single head, 80 tracks	35.00
5-1/4" single head, 80 tracks, flippy	345.00
5-1/4" double head, 80 X 2 tracks	470.00
ABPI - Service Manual	20.00
Siemens Manual	10.00
8" single head, 77 tracks	75.00
8" single head, 77 tracks, Thin-Line	450.00
8" double head, 77 X 2 tracks	495.00
8" double head, 77 X 2 tracks, Thin-Line	525.00
Microtime 6800 Calendar and Clock Board (assembled and tested)	105.00
Bareboard, connector, and documentation only of above (See review Feb 1980 '68 Micro Journal)	35.00
Microtime II	89.95
Date Mart 16K EPROM baseboard (270 chips)	30.00

Printers

Epson MX-80(Centronics compatible, parallel interface)	456.00
(with Serial RS-232 interface option)	add 75.00
Spare Print Head	39.95
Spare ribbon cartridge	15.00
C. Itoh Comet I 125 cps, 9 x 7, bidirectional, serial or parallel	425.00
Okidata Microline 82A 120 cps, 9 x 9 bidirectional, serial and parallel	495.00
Tractor for 82A	70.00
Optional Technology, Inc. EP-2A-79 Eprom Programmer (Personality Modules extra for above programmer)	169.00
Optional Technology, Inc. 30 pin parallel I/O board for EP-2A-79	35.00
Software package for EP-2A-79 (Specify 6800 or 6809)	29.95
EP-2A-78-M-01 Eprom Programmer (User configures and supplies power)	79.95

Smoke Signal Broadcasting

Monitor on 2716 with manual (Specify Ch 13 or SWTPC)	75.00
DCB-4A Double Density Controller Board for 5" and 8" with DOS	540.00
DOS69D DOS Update with Editor and Assembler (Specify 5" or 8")	75.00
LMB-1A Motherboard	300.00
SCB-69 6809 CPU Board	300.00
M-16-X 16K Static Memory Board	195.00
M-24-X 24K Static Memory Board	295.00
M-32-X 32K Static Memory Board	395.00

GIMIX

2 MHz 6809 Plus CPU, time of day clock, battery backup, 1K NMOS RAM	578.05
CMOS RAM substitution	8.00
GIMIX Dynamic Address Translator	35.00
SWPTC compatible DAT	15.00
Missing cycle detect card	38.23
Disk Controllers (All have data separators and can be used with either single or double headed drives)	
5" single density controller without 1771 chip	158.38
5" single density controller complete	198.48
5" and 8" single density controller complete	226.58
5" double density controller with variable precomp	298.28
DMA 5" AND 8" double density controller with variable precomp	588.68
GIMIX version of FLEX™ (without Editor and Assembler)	90.00
Double disk regulator card	68.22
Ribbon cable for two 5 1/4" disk drives (short)	34.96
Ribbon cable for two 5 1/4" disk drives (long)	44.96
Ribbon cable for two 8" disk drives (long)	44.26
8" disk drive cabinet with power supply	848.18

Memory	CMOS WITH BAT. BACKUP	NMOS NO BAT. BACKUP
16K Static RAM Board with control registers**		225.00
32K Static RAM Board with 32K of RAM installed*		350.00
**discontinued, limited quantity available		
64K Static RAM Board with 24K of RAM installed	N/A	348.27
64K Static RAM Board with 32K of RAM installed	518.36	398.37
64K Static RAM Board with 48K of RAM installed	N/A	518.47
64K Static RAM Board with 56K of RAM installed	728.56	578.57
64K Static RAM Board with 64K of RAM installed	798.64	638.67
16 Socket EPROM/ROM/RAM Board		238.32
8K Promboard (2708)		98.34
4K PPO 4K Prom Board and 2708 Prom Burner		198.00

I/O Boards

Single port 30 pin serial interface (Requires 1 cable set)	88.41
Dual port 30 pin serial interface (Requires 2 cable sets)	128.43
8 port 50 pin serial interface with baud rate generator	318.46
Dual port 30 pin parallel interface (Requires 2 cable sets)	88.42
8 port 50 pin parallel interface with interrupt generator	198.45
Cable sets for above boards (specify board)	24.95
2MHz 6809 PLUS Computer System with 56K Memory*	2498.29
Above System with 8SB Controller and Special Software Pkg.*	2988.59
Above System with 88B Controller and Special Software Pkg.*	3248.49
*with CMOS RAM and Battery Backup	add 300.00
Mainframe (Chassis, PS, Switches, Fan, Motherboard, Baud Rate Gen.)	1198.18

Shipping and handling estimates

Within the Continental U.S., please add 3% ($5.00 minimum)
Foreign, prepaid and add 10% ($10.00 minimum) for light items only.
Heavy items must be PrePaid and will be shipped Emery Air Freight Collect.

Please phone during consultation hours if questions arise regarding shipping fees.

Master Charge, Visa, and American ExPress honored

AAA Chicago Computer Center **Closed June 26 thru July 18**

120 Chestnut Lane, Wheeling, IL 60090

(312) 459-0450 Phone Consultation available most weekdays from 4 PM to 8 PM

Our apology. We are running 4 weeks behind answering our mail regarding technical questions. Please phone during consultation hours for technical help.

Prices and inventory are subject to change without advance notice.
This ad is our catalog.

UNIFLEX* SOFTWARE

by

Uni Comp . . .

ACCOUNTS PAYABLE

UniComp Accounts Payable is designed to enable complete control over a business' obligations. The program automatically ages invoices, and generates checks.

- Maintains a detailed vendor file.
- Generates comprehensive reports instantly to facilitate cash flow management.
- Completely menu driven requiring no special programming knowledge or experience.
- Interactive with UniComp General Ledger.

$595

ACCOUNTS RECEIVABLE

UniComp Accounts Receivable is designed to enable complete control over a business' receivables. The program automatically ages accounts, and generates statements.

- Maintains a detailed vendor file.
- Generates comprehensive reports instantly to facilitate cash flow management.
- Is completely menu driven requiring no special programming knowledge or experience.
- Is interactive with UniComp General Ledger.

$595

GENERAL LEDGER

The UniComp General Ledger package is a comprehensive program enabling a business to completely integrate and automate its accounting package.

- Designed to accomodate the entire ledger function and uses Generally Accepted Accounting Principles.
- Supports multiple companies/profit centers on the same diskette.
- Provides an optional full restriction feature.
- Is fully interactive with other UniComp accounting packages.

$995

INVENTORY/ ORDER ENTRY

The UniComp Inventory/Order Entry software package is a comprehensive program designed for global resource management. Its purpose is to integrate and automate all aspects of inventory/order entry processing.

- Generates purchase orders and reports, as well as vendor information files.
- Provides cash register operation enabling real time inventory update capabilities.
- Completely menu driven, with easy to read and understand instructions.

$995

UNIBASE

UniBase is a stand alone data base management program enabling users to design their own data base program. It is menu driven with no programming knowledge or experience necessary to use it.

- Allows up to 10 key fields, user filled.
- Automatically computes total record length and maximum number of records available based on storage available.
- May be sorted on any field or any combinations of fields.
- Allows restrictions on data access by terminal.

$795

MAILING LIST

The UniComp Mailing List program is designed to enable a business to handle all aspects of mailing list processing.

- Enable the user to maintain accurate customer/client lists and update them at any time.
- Automatically sorts records in alphabetic order or zipcode order.
- User may elect to sort records using any combination of parameters.
- Supports the printing of mailing labels—one, two, or three across, user designated.

$495

PAYROLL

UniComp Payroll is designed to completely automate all aspects of payroll processing.

- It generates payroll checks and stubs, internal reports, governmental reports, W2 and other reports.
- Enables the user to maintain extensive (70 fields) employee files.
- Completely interactive with the UniComp General Ledger package.
- Supports multiple companies/subsidiaries on the same diskette.

$795

Minimum Equipment Requirements

Computer/Display

A South West Technical Products computer with a minimum of 128K bytes of memory, 2 eight inch floppy double density, double sided disk drives, and at least on SWTPC 8212 or 8212W terminal.

Printer

Any ASCII compatible printer with at least a 132 column print width.

To Order Send Check or Money Order To:

COMPUWORLD Inc.

125 WHITE SPRUCE BLVD. • ROCHESTER, NY 14623 • (716) 424-6260

Unicomp is a trademark of Compuworld Inc.

Uniflex is a trademark of Technical Systems Consultants

VISA

For further information call:

716-424-6260

'68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

THE CHIEFTAIN™ 5¼-INCH WINCHESTER HARD DISK COMPUTER

SO ADVANCED IN SO MANY WAYS . . . AND SO COST-EFFECTIVE . . . IT OBSOLETES MOST OTHER SYSTEMS AVAILABLE TODAY AT ANY PRICE.

● HARD DISK SYSTEM CAPACITY

The Chieftain series includes 5¼- and 8-inch Winchesters that range from 4- to 60-megabyte capacity, and higher as technology advances. All hard disk Chieftains include 64-k memory with two serial ports and DOS69D disk operating system.

● LIGHTNING ACCESS TIME

Average access time for 5¼-inch Winchesters is 70-msec. comparable to far more costly hard disk systems. That means data transfer *ten-times faster* than floppy disk systems.

● 2-MHZ OPERATION

All Chieftains operate at 2-MHz, regardless of disk storage type or operating system used. Compare this to other hard disk systems, no matter *how* much they cost!

● DMA DATA TRANSFER

DMA data transfer to-and-from tape and disk is provided for optimum speed. A special design technique eliminates the necessity of halting the processor to wait for data which normally transfers at a slower speed, determined by the rotational velocity of the disk.

● RUNS UNDER DOS OR OS-9

No matter which Chieftain you select . . . 5¼- or 8-inch floppy, or 5¼- or 8-inch Winchester with tape or floppy back-up . . . they *all* run under DOS or OS-9 with *no need* to modify hardware or software.

● UNBOUNDED FLEXIBILITY

You'll probably never use it, but any Chieftain hard disk system can drive up to 20 other Winchesters, and four tape drives, with a single DMA interface board!

● SMOKE SIGNAL'S HERITAGE OF EXCELLENCE

This new-generation computer is accompanied by the same *Endurance-Certified* quality Dealers and end-users all over the world have come to expect from Smoke Signal. And support, software selection and extremely competitive pricing are very much a part of that enviable reputation.

20-Megabyte Tape Streamer Back-Up Option

Available with all Chieftain hard disk configurations. This cartridge tape capability provides full 20-megabyte disk back-up in less than five minutes with just one command, or copy command for individual file transfers. Transfers data tape-to-disk or disk-to-tape. Floppy back-up is also available in a variety of configurations.

The Chieftain Computer Systems:

Here are the Chieftain 6809-based hard disk computers that are destined to change the data processing industry . . .

☐ **CHIEFTAIN 95W4**
4-megabyte, 5¼-inch Winchester with a 360-k floppy disk drive (pictured).

☐ **CHIEFTAIN 95XW4**
4-megabyte, 5¼-inch Winchester with a 750-k octo-density floppy disk drive.

☐ **CHIEFTAIN 98W15**
15-megabyte, 5¼-inch Winchester with a 1-megabyte 8-inch floppy disk drive.

☐ **CHIEFTAIN 9W15T20**
15-megabyte, 5¼-inch Winchester with a 20-megabyte tape streamer.

Write or call today for details (including the remarkably low prices) on the total Chieftain Series . . . and on dealership opportunities.

SMOKE SIGNAL BROADCASTING®
31336 VIA COLINAS
WESTLAKE VILLAGE, CA 91362
TEL (213) 889-9340

Name _____
Company _____
Address _____
City _____ State _____ Zip _____
Telephone ( ) _____

JUDGE THE REST, THEN BUY THE BEST

Only GIMIX offers you **SOFTWARE SWITCHING** between **MICROWARE's OS-9** and **TSC's FLEX.** Plus you get the power of the GMXBUG system monitor with its advanced debugging utility, and memory manipulation routines. A wide variety of languages and other software is available for these two predominant 6809 Disk Operating Systems.

You can order a system to meet your needs, or select from the 6809 Systems featured below.

JUDGE THE FEATURES AND QUALITY OF GIMIX 6809 SYSTEMS

GIMIX' CLASSY CHASSIS™ is a heavyweight aluminum mainframe cabinet with back panel cutouts to conveniently connect your terminals, printers, drives, monitors, etc. A 3 position keyswitch lets you lock out the reset switch. The power supply features a ferro-resonant constant voltage transformer that supplies 8V at 30 amps, + 15V at 5 amps, and – 15V at 5 amps to insure against problems caused by adverse power input conditions. It supplies power for all the boards in a fully loaded system plus two 5 ¼" drives (yes! even a Winchester) that can be installed in the cabinet. The Mother board has fifteen 50 pin and eight 30 pin slots to give you the most room for expansion of any SS50 system available. 11 standard baud rates from 75 to 38.4K are provided and the I/O section has its own extended addressing to permit the maximum memory address space to be used. The 2 Mhz 6809 CPU card has both a time of day clock with battery back-up and a 6840 programmable timer. It also contains 1K RAM, 4 PROM/ROM/RAM sockets, and provides for an optional 9511A or 9512 Arithmetic Processor. The RAM boards use high speed, low power STATIC memory that is fully compatible with any DMA technique. STATIC RAM requires no refresh timing, no wait states or clock stretching, and allows fast, reliable operation. The system includes a 2 port RS232 serial interface and cables. All GIMIX boards use gold plated bus connectors and are fully socketed. GIMIX designs, manufactures, and tests in-house its complete line of products. All boards are twice tested, and burned in electrically to insure reliability and freedom from infant mortality of component parts. All systems are assembled and then retested as a system after being configured to your specific order.

56KB 2MHZ 6809 SYSTEMS WITH GMXBUX/FLEX/OS-9 SOFTWARE SELECTABLE

With #58 single density disk controller **$2988.59**
With #68 DMA double density disk controller **$3248.49**
to substitute Non-volatile CMOS RAM with battery back-up, add **150.00**
for 50 Hz export power supply models, add **30.00**

Either controller can be used with any combination of 5" and/or 8" drives, up to 4 drives total, have data recovery circuits (data separators), and are designed to fully meet the timing requirements of the controller I.C.s.

5 ¼" DRIVES INSTALLED IN THE ABOVE with all necessary cables

	SINGLE DENSITY		DOUBLE DENSITY		
	Formatted	Unformatted	Formatted	Unformatted	
40 track (48TPI) single sided	199,680	250,000	341,424	500,000	2 for **$700.00**
40 track (48TPI) double sided	399,360	500,000	718,848	1,000,000	2 for **900.00**
80 track (96TPI) single	404,480	500,000	728,064	1,000,000	2 for **900.00**
80 track (96TPI) double	808,960	1,000,000	1,456,128	2,000,000	2 for **1300.00**

Chart shows total capacity in Bytes for 2 drives.

Contact GIMIX for price and availability of 8" floppy disk drives and cabinets; and 5" and 8" Winchester hard disk system.

128KB 2Mhz 8809 DMA Systems for use with TSC's UNIFLEX or MICROWARES's OS-9 Level 2

(Software and drives not included) **$3798.39**
to substitute 128KB CMOS RAM with battery back-up, add **300.00**
for each additional 64KB NMOS STATIC RAM board, add **638.67**
for each additional 64KB CMOS STATIC RAM board, add **798.64**
for 50 Hz export power supply, add **30.00**

NOTE: UNIFLEX can not be used with 5" minifloppy drives.

GIMIX has a wide variety of RAM, ROM, Serial and Parallel I/O, Video, Graphics, and other SS50 bus cards that can be added now or in the future. Phone or write for more complete information and brochure.

THE SUN NEVER SETS ON GIMIX USERS

GIMIX Systems are found on every continent, except Antarctica. (Any users there? If so, please contact GIMIX so we can change this.) A representative group of GIMIX users includes: **Government Research and Scientific Organizations** in Australia, Canada, U.K., and in the U.S.: NASA, Oak Ridge, White Plains, Fermilab, Argonne, Scripps, Sloan Kettering, Los Alamos National Labs, AURA. **Universities**: Carleton, Waterloo, Royal Military College, in Canada; Trier in Germany; and in the U.S.: Stanford, SUNY, Harvard, UCSD, Mississippi, Georgia Tech. **Industrial users** in Hong Kong, Malaysia, South Africa, Germany, Sweden, and in the U.S.: GTE, Becton Dickinson, American Hoechst, Monsanto, Allied, Honeywell, Perkin Elmer, Johnson Controls, Associated Press, Aydin, Newkirk Electric, Revere Sugar, HI-G/AMS Controls, Chevron. **Computer mainframe and peripheral manufacturers**: IBM, OKI, Computer Peripherals Inc., Qume, Floating Point Systems. **Software houses**: Microware, T.S.C., Lucidata, Norpak, Talbot, Stylo Systems, AAA, HHH, Frank Hogg Labs, Epstein Associates, Softwest, Dynasoft, Research Resources U.K., Microworks, Analog Systems, Computerized Business Systems.

TO ORDER BY MAIL

SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $600.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, account #72-32003. Visa or Master Charge also accepted.

GIMIX INC. reserves the right to change pricing and product specifications at any time without further notice.

GIMIX Systems are chosen by the Pros because of quality, reliability and features.

GIMIX INC.

The Company that delivers
Quality Electronic products since 1975.

1337 WEST 37th PLACE, CHICAGO, IL 60609
(312) 927-5510 • TWX 910-221-4055

E X T R A
FLEX & RS COLOR COMPUTER

With the EXATRON INTERFACE and DISK CONTROLLER

GOOD NEWS #1 — The popular, easy to use, and very powerful FLEX9 Disk Operating System has been running on the Radio Shack Color Computer for several months using the Special EXATRON 32K Interface and Disk Controller. This System adds 32K Memory to a 16K Color Computer, providing 48K of RAM for User Programs, the FLEX Disk Operating System, etc. This is accomplished by adding 16K more Memory between the 16K in the Color Computer and the BASIC ROMs, and adding 16K of Memory ABOVE the BASIC ROMs (which is REQUIRED for the FLEX Disk Operating System). This System requires NO MODIFICATIONS to the Computer. It operates as a Single Sided, SINGLE DENSITY, 35 Track, System, and controls up to 4 Disk Drives. Most MAJOR Programs must be modified to run on this System.

F-MATE(EX) — This is a set of SPECIAL SOFTWARE ROUTINES supplied on a 5 1/4" Diskette which "Installs" the Special General FLEX Operating System on the Radio Shack COLOR COMPUTER as a "bootable" System for normal use. The normal FLEX9 Utilities such as COPY, CATalog, LIST, RENAME, DELETE, etc. are included to provide a fully usable Disk Operating System.

PATCHES — To patch and make functional various standard TSC Utilities and programs including APPEND, ASMB, EDIT, PUTLDR, SAVE.LOW, XBASIC, and others. A special NEWDISK routine allows disks made on the COLOR COMPUTER to be read or written on other FLEX9 Systems.

- - - - SYSTEM REQUIREMENTS - - - -

FLEX9 Special General Version including the Editor and Assembler (NOTE: the Editor and Assembler each sell for $50.00, so you get FLEX9 for $50.00). $150.00

Special EXATRON 32K Expansion/Disk Controller $299.95

F-MATE(EX) FLEX9 Conversion for EXATRON SYS.
when purchased with Spec. FLEX9 Sys. $49.95
when purchased without Spec. FLEX9 Sys. $59.95

Screen-Clean - R.F. Noise Eliminator for EXATRON SYSTEM Wired and Tested $39.95

Radio Shack 16K COLOR COMPUTER, with Extended BASIC and ready for the above items $499.95

- - - FOR USERS THAT ALREADY HAVE FLEX9 & Disk Drives - - -

Radio Shack DISK CONTROLLER with F-MATE(RS) and a Special Two Drive DISK CABLE $289.95

- - - DISK DRIVE PACKAGES, with RS Controller - - -

These Packages include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable:

PAK #1 ==> 1 Single Sided, Double Density Sys. $529.95
PAK #2 ==> 2 Single Sided, Double Density Sys. $779.95
PAK #3 ==> 1 Double Sided, Double Density Sys. $649.95
PAK #4 ==> 2 Double Sided, Double Density Sys. $949.95

- - - - - - - PARTS AND PIECES - - - - - - -

Radio Shack Disk Controller	$199.95
1 ea. Single Sided, Double Density Disk Drive	$249.95
1 ea. Double Sided, Double Density Disk Drive	$349.95
Single Drive Cabinet with Power Supply	$79.95
Double Drive Cabinet with Power Supply	$99.95
Single Drive Disk Cable for RS Controller	$24.95
Double Drive Disk Cable for RS Controller	$34.95
Micro Tech. Prods., Inc. LOWER CASE ROM Adapter	$74.95
Radio Shack BASIC Version 1.1 ROM	$34.95

VISA or

MASTER CHARGE accepted

Add $25.00 Shipping & Handling For Complete Set

With the RADIO SHACK COLOR COMPUTER DISK CONTROLLER

GOOD NEWS #2 — The popular, easy to use, and very powerful FLEX9 Disk Operating System is AVAILABLE NOW for the Radio Shack COLOR COMPUTER with the RADIO SHACK COLOR COMPUTER DISK CONTROLLER. This system requires a Version 1.1 BASIC ROM and 64K RAM. This is easily accomplished on a normal Radio Shack 32K Color Computer, which already has the Version 1.1 ROM and memory bank select jumpers, by replacing the existing RAM Chips with KNOWN GOOD 64K Chips and enabling one NOR gate. If you do not have a 32K System, you can have it updated by a Radio Shack Service Center, or purchase a Version 1.1 ROM and modify it yourself. Data Comp can supply GUARANTEED 64K Memory Chips and instructions for the modification (see below).

F-MATE(RS) — This is a set of SPECIAL SOFTWARE ROUTINES supplied on TWO 5 1/4" Disks which provide the conversion routines for developing a normal "bootable" FLEX9 System for operation WITH THE RADIO SHACK COLOR DISK CONTROLLER. The F-MATE(RS) "INSTALLATION" Disk contains the routines which "marry" the FLEX9 Disk Operating System to the specific requirements of the Radio Shack Color Computer. The F-MATE(RS) "UTILITIES" Disk contains the Special Routines developed for this Package and accomplish the normal Input/Output conversions, along with the necessary Software to activate the 64K Memory System, relocate the Display Screen Memory and variable areas, provide NEW, INDEPENDENT KEYBOARD and DISPLAY CAPABILITIES, etc.

FEATURES — Data-Comp's F-MATE(RS) DOES NOT REQUIRE A "PATCHES" CONVERSION. This adaptation allows ALL FLEX9 Compatible Software which uses the normal FLEX9 I/O routines to run on the Radio Shack COLOR COMPUTER WITHOUT MODIFICATION. Special COLOR COMPUTER Utilities supplied include:

1. FIVE different DISPLAY SCREENS (supplied with the Source Code so you can develop your own character set). 32 x 16 (the normal CC Screen), 32 x 24, 42 x 24, 51 x 24, and 64 x 24 Display Screens are available via a simple system command.
2. SAVE ROMs - a routine which allows saving the BASIC ROM's to a FLEX9 Disk, so normal Radio Shack BASIC can be called and run with the 64K Memory still enabled.
3. DISK and MEMORY Diagnostic Routines.
4. EXTENDED KEYBOARD including full "CONTROL" Key functions, an "ESCAPE" Key, and 12 user definable keys.
5. SPECIAL NEWDISK Routine for Formatting Single or Double SIDE, Single or Double DENSITY, 35, 40 or 80 TRACK Diskettes.
6. System capable of running up to THREE DOUBLE SIDED DRIVES, or FOUR DRIVES if none are Double Sided.

- - - - SYSTEM REQUIREMENTS - - - -

FLEX9 Special General Version w/ Editor and Assembler (the EDITOR and ASSEMBLER normally sell for $50.00 each, so you get Gen. FLEX for $50.00) $150.00

F-MATE(RS) FLEX9 Conversion for R. S. Disk System
when purchased with Spec. FLEX9 Sys. $49.95
when purchased without Spec. FLEX9 $59.95

Set of eight 64K RAM Chips w/ mod instructions $99.95

64K RAM Radio Shack COLOR COMPUTER System $649.95

- - - - - - SPECIAL SYSTEM PACKAGES - - - - - -

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9, F-MATE(RS), and a Box of 10 Double Density Diskettes; a COMPLETE, ready to run SYSTEM on your Color TV Set $1379.95

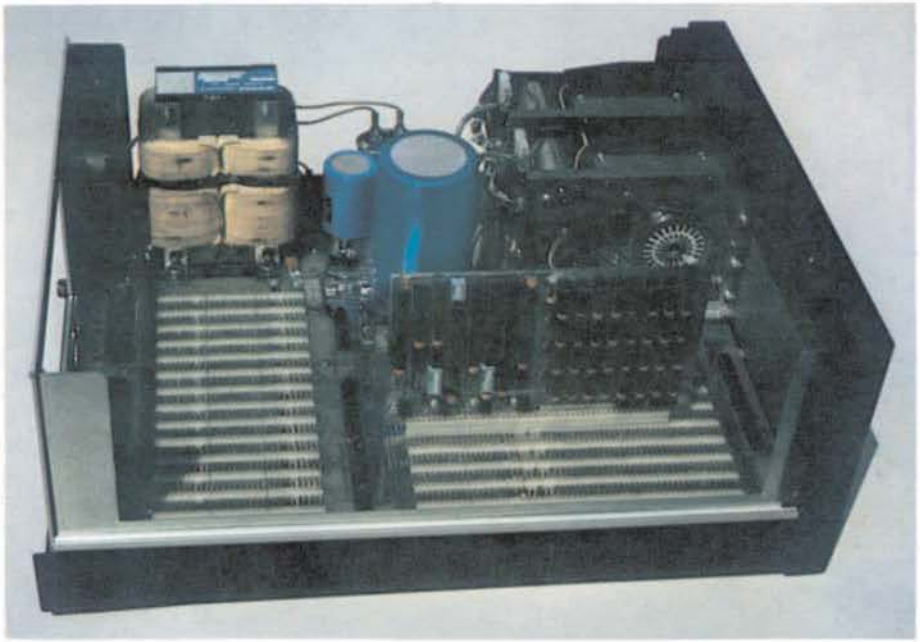
DATA-COMP
SOUTH EAST MEDIA
P.O. Box 794 Chattanooga TN 37443

1-615-842-4601

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

HELIX™



THE MAINFRAME

- Industry Standard Optima™ Cabinet
- Largest Constant Voltage Power Supply in the Industry
- S-64 Bus gives 16 Bit Power and S-50 Bus Compatibility
- 10 Main (S-64) Slots
- 14 I/O (S-30) Slots plus 2 On-board
- On-board Baud Rate Generator to 38.4Kb
- Space and Power for two 5¼" Disk Drives
- Full Address Decoding for I/O Slots
- Two RS-232 Serial and Two parallel Ports On-board
- Single Board Construction for Reliability
- Faraday Shielded Bus Lines give "Text Book Clean" Signals

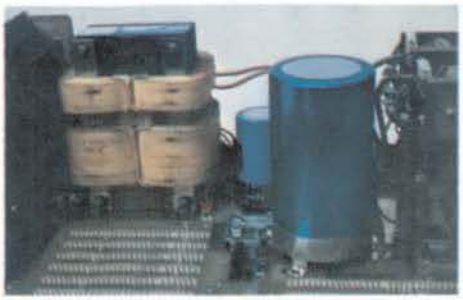
THE PROCESSORS

6809

- Standard 2 MHz Operation
- Standard DAT Compatible with GIMIX and SWTPC
- Standard 6840 Interval Timer
- Standard 1K Scratchpad RAM
- Standard Clock/Calendar with Battery
- Provision for Programmers Console

68000

- Standard 8 MHz Operation
- Memory Management Hardware
- Provision for Programmers Console
- 16 Bit Power and 8 Bit Compatibility



The HELIX™ computer system represents the latest advance in S-50 bus computer systems. Relying on the physical nature of S-50 bus connectors to guarantee compatibility, the HELIX adds 14 bus lines (becoming S-64) to allow a 68000 processor to operate with full 16 bit data transfer and 24 bit addressing, while at the same time providing full interchangability with existing S-50 components.

Offered with a selection of processors, memories, and peripheral controllers, a HELIX system can be configured for applications ranging from advanced hobbyist to multiterminal time-sharing.

Designed to offer the utmost in speed, reliability, and utility at a reasonable price, it represents a new standard of quality for those who require a professionally designed computer for professional use.

THE POWER SUPPLY

- Ferro-resonant Transformer for Line Noise and Under-Voltage Protection
- Conservative 25 Amps at 8.5 Volts
- Conservative 5 Amps at ±16 Volts
- Conservative Component Rating for Reliability

THE COMPONENTS

- Fully Socketed
- Gold Plated Bus Connectors
- Only "B" Series 68XX Components Used
- Only Top Grade Logic Circuits Used
- Industrial Grade Components Throughout

THE MEMORIES

DM-64

- Field Proven
- Proprietary Memory Control Logic
- Fully Transparent Refresh
- Tested at 2.5 MHz Operation

DM-512

- 512K Bytes on a Single S-64 Board
- 16 Bit Power and 8 Bit Compatibility
- Runs in Existing S-50 Systems where Physical Space Allows
- Full 24 Bit Addressing
- Fully Transparent Refresh

THE PRICES

Because of the variety of configurations possible, full pricing cannot be given. Representative prices are:

- 64K 6809 HELIX $1995
- 64K 68000 HELIX $2595
- 512K 6809 HELIX $4450
- 512K 68000 HELIX $4995

HAZELWOOD COMPUTER SYSTEMS

7413 N. Lindbergh, Hazelwood, Missouri 63042 (314) 281-1055

Dealer and OEM Inquiries Invited. We support our Dealers.

Optima is a Trademark of Scientific-Atlanta